FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	SOS - Studentsky odevzdavaci system
Student:	Tomáš Pavlůsek
Vedoucí:	Ing. Jiří Hunka
Studijní program:	Informatika
Obor / specializace:	Webové a softwarové inženýrství, zaměření Softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce letního semestru 2022/2023

Pokyny pro vypracování

Cílem práce je návrh a realizace webového systému pro snadné odevzdávání týmových i samostatných semestrálních prací.

Postupujte dle následujících kroků:
Analyzujte potřeby předmětu MI-NUR z pohledu tvorby a odevzdávání týmových semestrálních prací. Zaměřte se i na předchozí systém NURIS. Dále analyzujte možnosti vyhovující i pro jiné předměty, kde je cílem realizovat a odevzdat semestrální práci po částech, či najednou.
S ohledem na předešlou analýzu proveďte řádný návrh potřebného řešení.
Na základě návrhu realizujte funkční prototyp budoucího webového systému.
Prototyp podrobte vhodnému testování, které sám navrhnete.
Na základě zjištěných nedostatků realizujte finální webovou aplikaci.
Shrňte dosažené výsledky, navrhněte budoucí rozšíření či směřování projektu.

Elektronicky schválil/a Ing. Michal Valenta, Ph.D. dne 11. března 2021 v Praze.

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

SOS - Studentský odevzdávací systém

Tomáš Pavlůsek

Katedra softwarového inženýrství
Vedoucí práce: Ing. Jiří Hunka

13. května 2021

Poděkování

Chtěl bych poděkovat svému vedoucímu bakalářské práce Ing. Jiřímu Hunkovi za možnost zvolit si toto téma a za odborné vedení, trpělivost a ochotu, kterou mi v průběhu zpracování této práce věnoval.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 2373 odst. 2 zákona č. 89/2012 Sb., občanský zákoník, ve znění pozdějších předpisů, tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu) licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 13. května 2021 . . . . . .

České vysoké učení technické v Praze
Fakulta informačních technologií

Odkaz na tuto práci

Pavlůsek, Tomáš. *SOS - Studentský odevzdávací systém.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2021. Dostupný také z WWW: ⟨`https://sos.fit.cvut.cz/`⟩.

Abstrakt

Tato bakalářská práce se zabývá vývojem webové aplikace SOS – Studentský odevzdávací systém. SOS je informační systém sloužící k odevzdávání a hodnocení týmových i individuálních semestrálních prací v předmětech vyučovaných na Fakultě informačních technologií Českého vysokého učení technického v Praze. Analytická část práce se zabývá analýzou potřeb studentů i vyučujících v těchto předmětech a návrhem potřebného řešení, včetně volby vhodných technologií. Realizační část práce zahrnuje kompletní implementaci webové aplikace s využitím webového aplikačního frameworku Django, integraci systému s dalšími informačními systémy využívanými na Fakultě informačních technologií a nasazení systému do provozu na fakultní server s využitím CI/CD. Na závěr je popsán proces testování aplikace jejími potenciálními uživateli a reakce na takto získanou zpětnou vazbu. Výsledkem práce je funkční software, který je možné nasadit do testovacího provozu při výuce na Fakultě informačních technologií.

Klíčová slova odevzdávací systém, systém pro správu týmů, systém pro podporu výuky, webová aplikace, Django, vícestránková aplikace

Abstract

The subject of this bachelor thesis is development of the web application SOS – Student submission system. SOS is an information system used for submission and evaluation of both team and individual semestral works in courses taught at the Faculty of Information Technology at the Czech Technical Univirsity in Prague. The subject of the analytical section of the thesis is the analysis of requirements of both students and teachers in said courses and design of the required solution, including the choice of appropriate technologies. The implementation section of the thesis includes complete implementation of the web application using the web application framework Django, integration with other information systems used at the Faculty of Information Technology and deployment of the system on the faculty server using CI/CD. Finally, the process of testing the application with its potential users is described, along with the response to the feedback thus obtained. The result of the thesis is a functional software, which can be deployed into test operation in teaching at the Faculty of Information Technology.

Keywords submission system, team management system, teaching support system, web application, Django, multi-page application

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Vzhledem k počtu studentů neustále roste potřeba využití softwarových systémů pro správu výuky. Stejně tomu je i u předmětů vyučovaných na Fakultě informačních technologií Českého vysokého učení technického v Praze. Vyučuje se zde množství předmětů, u kterých se podstatná část jejich náplně zakládá na vypracování semestrální práce, ať už samostatně nebo v týmech. Namátkou jsou to předměty MI-NUR: Návrh uživatelského rozhraní, BI-OOP: Objektově orientované programování nebo BI-SP1: Softwarový týmový projekt 1 a BI-SP2: Softwarový týmový projekt 2.

V současné době neexistuje jednotný odevzdávací systém, který by naplňoval potřeby jednotlivých předmětů. Jsou zde sice univerzální systémy, jako je například Moodle, ten ale není příliš vhodný pro odevzdávání softwarových projektů. Také nepodporuje správu projektů a týmů přímo studenty. Týmy se je tak nutno vytvářet na cvičeních nebo zdlouhavou e-mailovou komunikací. Pro některé předměty sice již existují specifická řešení, není ale příliš efektivní implementovat a udržovat samostatný systém pro každý takový předmět na fakultě, když jsou jejich potřeby často velmi podobné.

SOS bude univerzální systém pro správu týmů a odevzdávání a následné hodnocení týmových i individuálních projektů, konfigurovatelný pro jednotlivé předměty tak, aby vyhovoval jejich potřebám a způsobu hodnocení.

Práce navazuje na systém NURIS, který toto činí pro předmět MI-NUR. Tento systém ale není využitelný pro ostatní předměty a nelze jej jednoduše takto rozšířit. Proto bylo přistoupeno k implementaci zcela nového systému SOS, který NURIS nahradí a zároveň jej využijí i ostatní předměty, jako jsou například již výše zmíněné BI-OOP, BI-SP1 a další.

Toto téma bylo autorovi práce navrženo vedoucím práce Ing. Jiřím Hunkou. Autor byl během svého bakalářského studia na Fakultě iformačních technologií obeznámen s průběhem výuky a požadavky jednotlivých předmětů a má zkušenosti s vývojem webových aplikací. Má tedy dobré předpoklady vyvinout v rámci této práce systém, který bude pro fakultu užitečný.

Kapitola 1

Cíl práce

Cílem této bakalářské práce je navrhnout, naimplementovat a nasadit SOS - Studentský odevzdávací systém. Tento systém bude sloužit pro správu týmových projektů v předmětech vyučovaných na Fakultě informačních technologií Českého vysokého učení technického v Praze.

Práce navazuje na NURIS - Informační systém pro MI-NUR [1], lze tedy částečně využít poznatky ohledně potřeb studentů a vyučujících v MI-NUR z této práce. Cílem je ale také analyzovat i další předměty vyučované na Fakultě informačních technologií, kterým by mohl SOS potenciálně sloužit. Je nutné kompletně navrhnout a implementovat všechny části systému - databázi, backend a frontend. Dále také provést konfiguraci systému a nasazení na fakultní server.

Bude kladen důraz na udržitelnost a rozšiřitelnost systému, aby mohl dlouhodobě plnit požadavky co nejvíce předmětů, ve kterých studenti vypracovávají týmové či individuální projekty, vyučovaných na Fakultě infomačních technologií.

Dílčím cílem práce je i napojení systému na ostatní informační systémy používané na fakultě. Konkrétně se jedná o autentizaci uživatelů skrze fakultní OAauth 2.0 autorizační server a získávání dat o předmětech a uživatelích ze systému KOS prostřednictvím jeho API.

Kapitola 2

Analýza a návrh

Práce navazuje na systém NURIS, což je odevzdávací systém pro semestrální práce v předmětu MI-NUR: Návrh uživatelského rozhraní. Požadavky ze strany studentů i vyučujících v MI-NUR byly již analyzovány při vývoji systému NURIS a lze je tedy s drobnými aktualizacemi využít i zde. [1]

Hlavním předmětem analýzy jsou ale podobnosti a rozdíly mezi potřebami MI-NUR a dalších předmětů vyučovaných na Fakultě informačních technologií, ve kterých studenti odevzdávají semestrální práce. Při analýze potřeb jednotlivých předmětů autor vychází především z popisů předmětů v systémech Courses a Moodle, ale také ze svých vlastních zkušeností z bakalářského studia na Fakultě informačních technoligií, během kterého řadu z těchto předmětů sám absolvoval. Při analýze autor využívá metod softwarového inženýrství, jako jsou analýza požadavků a modelování případů užití.

Poté se autor zabývá typy známými typy architektur využívaných při vývoji informačních systémů a jejich porovnáním. Dále rozebírá jednotlivé části budoucího systému, jako je datové úložiště a jeho struktura, nebo uživatelské rozhraní. Nakonec nabízí přehled konkrétních technologií, které lze pro využít pro implementaci, srovnává je a zdůvodňuje svůj výběr technologií pro tuto práci.

2.1 Systém NURIS

Diplomová práce Davida Jirouta z roku 2019 [1] přinesla systém NURIS – Informační systém pro správu předmětu MI-NUR: Návrh uživatelského rozhraní. Tento systém je již nasazen a aktivně využíván při výuce předmětu MI-NUR. Je postaven na technologiích ASP.NET Core a Angular Universal. Systém během vývoje prošel i uživatelským testováním a je tedy realizován tak, aby vyhovoval jak požadavkům uvedeným v analytické části diplomové práce, tak i těm, které vyplynuly z následného testovacího provozu systému.

Systém je využitelný pro předmět MI-NUR, nicméně pro ostatní předměty

nikoliv. Také se jeví jako obtížně udržovatelný a rozšiřitelný, vzhledem k tomu, že je navržen specificky pro přemět MI-NUR a během jeho vývoje nebyla rozšiřitelnost pro využití v ostatních předmětech zohledněna.

2.2 Průběh předmětu MI-NUR: Návrh uživatelského rozhraní

Průběh předmětu MI-NUR byl již detailně analyzován v diplomové práci, z níž vzešel systém NURIS (konkrétně v sekci 3.2.) [1] a do současnosti se téměř nezměnil [2]. Pro úplnost je nicméně tato analýza ve stručnosti uvedena i zde.

V rámci předmětu MI-NUR studenti nejprve na začátku semestru vytvoří dvou až čtyřčlené týmy. Dále vymyslí téma semestrálního projektu a postupně na cvičeních ve třech iteracích odevzdávají svá vypracovaná řešení.

Za jednotlivá dílčí odevzdání jsou studenti odměněni body. Další body mohou získat za testování řešení jiných týmů. Za včasné odevzdání všech iterací a splnění minimálního počtu získaných bodů získají studenti právo na udělení zápočtu.

2.3 Potřeby dalších předmětů

Na Fakultě informačních technologií jsou vyučovány i další předměty, ve kterých studenti vypracovávají semestrální práce. Tento proces je často téměř totožný s procesem tvorby semestrální práce v předmětu MI-NUR. Mezi společné části patří především postupné odevzdávání práce v jednotlivích iteracích, potřeba realizace týmové spolupráce studentů a hodnocení odevzadných řešení vyučujícími.

Potřeby jednotlivých předmětů mají svá specifika, jako například počet a obsah jednotlivých kontrolních bodů, termíny odevzdání nebo počet studentů v týmu. V některých předmětech studenti vypracovávají své semestrální práce samostatně. V předmětu MI-NUR tvoří téma semestrální práce sami studenti, v jiných předmětech jej často určuje vyučující. V takovém případě také může být toto téma společné pro všechny studenty daného předmětu. Uživatelské testování projektů je zase specifikum předmětu MI-NUR a v jiných předmětech se nerealizuje.

V následujících třech podsekcích autor uvádí příklady jednotlivých předmětů vyučovaných na Fakultě informačních technologií a specifika jejich procesu tvorby semestrální práce. Z těchto příkladů jasně vyplývá, že proces i forma odevzdávání semestrální práce musí být ve všech ohledech konfigurovatelné jednotlivě pro každý předmět, má-li systém sloužit i jiným předmětům, než je pouze MI-NUR.

2.3.1 BI-OOP: Objektově orientované programování

V předmětu BI-OOP si studenti na začátku semestru zvolí jedno z pěti vyučujícím definovaných zadání, které chtějí implementovat. Semestrální práce vypracovávají samostatně. Není zde vyžadováno odevzdávání práce v pevně definovaných iteracích, odevzdává se až kompletní implementace. Vzhledem k tomu, že výsledkem semestrální práce je počítačový program ve formě zdrojového kódu, je vyžadováno odevzdání prostřednictvím repozitáře na fakultním GitLab serveru. Finální odevzdání projektu studenti provádí vyplněním webového odevzdávacího formuláře. [3]

U tohoto předmětu je nutno poznamenat, že jeho náplň a průběh byly v roce 2019 kompletně přepracovány a stále se mění. Když autor v zimním semestru roku 2019 studoval tento předmět, proces tvorby semestrální práce se od současného stavu lišil v několika aspektech:

- Všichni studenti vypracovávali jediné společné zadání definované vyučujícím.
- Práce probíhala v týmech složených ze třech až čtyřech studentů. Složení týmů bylo určeno náhodně.
- Proces vývoje byl rozdělen na tři pevně dané milníky se stanovenými termíny odevzdání.

[4] Vzhledem k těmto faktům lze i v blízké budoucnosti očekávat určitou proměnlivost průběhu tohoto předmětu. Má-li systém být využitelný pro předmět BI-OOP, musí být zcela jistě co nejvíce konfigurovatelný, aby mohl dobře reagovat na požadované změny průběhu předmětu.

2.3.2 BI-KOM: Konceptuální modelování

V tomto předmětu studenti vypracovávají své semestrální práce samostatně nebo ve dvojicích. Téma práce si definují studenti sami. Výsledkem práce je PDF dokument, který je jedním ze studentů nahrán do systému Moodle. V případě, že je práce realizována ve dvojici, musí druhý člen týmu v systému Moodle odevzdání práce potvrdit. Práce je odevzdávána ve 2 iteracích se stanovenými termíny odevzdání. [5]

2.3.3 BI-SP1: Softwarový týmový projekt 1 a BI-SP2: Softwarový týmový projekt 2

Předměty BI-SP1 a BI-SP2 autor pro účely této analýzy spojil dohromady, vzhledem k tomu, že na sebe navazují a mají téměř totožný průběh. Studenti většinou v BI-SP2 dále rozvíjejí projekt, na kterém pracovali v BI-SP1.

V těchto předmětech vypracovává každý tým unikátní zadání. Zadání projektů jsou určena vyučujícími a následně zveřejněna studentům, kteří následně

projevují zájem o účast na daném projektu. Vyučující pak tyto studenty rozdělí do týmů o čtyřech až šesti členech.

Výsledkem semestrálních prací je softwarový projekt se všemi jeho náležitostmi, jako je dokumentace a zdrojové kódy. Mezi využívanou infrastrukturu patří především repozitáře na fakultním GitLab serveru, jak pro správu verzí zdrojového kódu, tak i pro ukládání dokumentace, poznámek a dalších dokumentů souvisejících s projektem.

Hlavní specifikum těchto předmětů je jejich způsob hodnocení. Práce je většinou vyhodnocována ve třech iteracích, nicméně počet iterací i jejich termíny odevzdání se mohou u jednotlivých vyučujících lišit. Za každou iteraci obdrží tým od vyučujícího určitý počet bodů. Vedoucí týmu tyto body následně podle stanovených pravidel přerozdělí mezi členy týmu tak, aby bodový zisk každého člena týmu odpovídal jeho aktivitě a odvedené práci. [6][7]

2.4 Analýza požadavků

Analýza požadavků má obecně za cíl vymezit hranice systému a zachytit omezení, která jsou na systém kladena. V praxi umožňují dodavateli vyjasnit si před začátkem vývoje systému zadání se zákazníkem a tím zajistit, že bude dodáno to, co zákazník skutečně potřeboval či požadoval. Při vývoji komerčních softwarových projektů je také rozsah díla daný specifikací požadavků parametrem při určovaní ceny za vývoj tohoto díla. Definované požadavky na systém taktéž umožňují provedení přesnějšího odhadu pracnosti vývoje a slouží jako základ pro vytváření dokumentace.

Správně definovaný požadavek musí musí být definovaný tak, aby jej nebylo možné interpretovat více různými způsoby. Musí být splnitelný, nesmí si tedy navzájem odporovat s ostatními požadavky. Je také potřeba, aby požadovanou funkcionalitu či vlastnost byl dodavatel systému schopen zajistit. Nakonec musí být možné splnění požadavku jednoznačně ověřit. Součástí definice požadavku je také určení náročnosti a důležitosti jeho naplnění. [8]

Požadavky lze rozdělit do dvou základních kategorií – funkční a nefunkční. Funkční požadavky popisují jednotlivé funkcionality a schopnosti systému. Nefunkční požadavky udávají omezení kladená na systém a vlastnosti systému jako takového. Přesnější kategorizaci požadavků umožňuje například metodika FURPS [9], která definuje tyto kategorie požadavků:

- **Functionality** (funkčnost) – funkční požadavky
- **Usability** (použitelnost) – jak lze systém používat, jak systém působí na uživatele, požadavky na dokumentaci
- **Reliability** (spolehlivost) – dostupnost systému, četnost a závažnost chyb a výpadků
- **Performance** (výkon) – rychlost odezev systému a technické parametry

- **Supportability** (podporovatelnost/rozšiřitelnost) – údržba a podpora systému, budoucí možnost rozšíření o další vlastnosti a přízpůsobitelnost systému novým okolnostem

2.4.1 Funkční požadavky

Z analýzy průběhu předmětu průběhu MI-NUR, dalších předmětů s podobným průběhem a jejich rozdílů a podobností vyplývají následující funkční požadavky:

- **FP1 – Konfigurace předmětů**: Aplikace umožňuje vytvoření profilů pro jednotlivé předměty a jejich následnou konfiguraci tak, aby odpovídaly průběhu těchto předmětů.
  Důležitost: Vysoká
  Náročnost: Vysoká

- **FP2 – Načítání dat**: Aplikace umožňuje načítat data o předmětech a jejich studentech a vyučujících ze systému KOS.
  Důležitost: Vysoká
  Náročnost: Střední

- **FP3 – Definice témat semestrálních prací**: Studenti a vyučující v aplikaci mohou definovat témata semestrálních prací.
  Důležitost: Vysoká
  Náročnost: Nízká

- **FP4 – Tvorba týmů**: Aplikace umožňuje zformovat týmy, ve kterých budou studenti pracovat a přiřadit jim téma, které vypracují.
  Důležitost: Vysoká
  Náročnost: Vysoká

- **FP5 – Odevzdávání řešení**: Studenti mohou v iteracích odevzdávat svá vypracovaná řešení včetně příloh v podobě nahraných souborů a webových odkazů.
  Důležitost: Vysoká
  Náročnost: Střední

- **FP6 – Uživatelské testování**: Studenti mohou v aplikaci potvrzovat, že jejich vypracovaná řešení byla otestována jinými studenty.
  Důležitost: Střední
  Náročnost: Střední

- **FP7 – Hodnocení**: Vyučující může v aplikaci provádět hodnocení odevzdaných řešení.
  Důležitost: Vysoká
  Náročnost: Nízká

2.4.2 Nefunkční požadavky

Z primárního účelu aplikace a jejího očekávaného způsobu používání vyplývají následující nefunkční požadavky:

- **NP1 – Dostupnost**: Aplikace je dostupná na fakultním serveru 24 hodin denně, 7 dní v týdnu.
  Důležitost: Vysoká
  Náročnost: Střední
  Kategorie: Spolehlivost

- **NP2 – Opřávnění pro přístup**: Aplikace a její data jsou dostupná pouze oprávněným uživatelům. Tito uživatelé do aplikace získávájí přístup prostřednictvím fakultního autentizačního serveru OAuth 2.0.
  Důležitost: Vysoká
  Náročnost: Střední
  Kategorie: Spolehlivost

- **NP3 – Doba odezvy**: Maximální doba reakce systému na požadavek je 10 sekund. 90 % požadavků je obslouženo do 500 milisekund za předpokladu, že má uživatel k dispozici internetové připojení odpovídající kvality.
  Důležitost: Střední
  Náročnost: Střední
  Kategorie: Výkon

- **NP4 – Ovládání aplikace**: Aplikaci je možné ovládat pomocí grafického uživatelského rozhraní v běžném webovém prohlížeči.
  Důležitost: Vysoká
  Náročnost: Střední
  Kategorie: Použitelnost

- **NP5 – Responzivita**: Aplikace má responzivní uživatelské rozhraní, je ji tedy možné pohodlně používat jak na běžném počítači, tak i na mobilních zařízeních.
  Důležitost: Střední
  Náročnost: Střední
  Kategorie: Použitelnost

- **NP6 – Jazyková lokalizace**: Uživatelské rozhraní aplikace je dostupné v českém a anglickém jazyce.
  Důležitost: Střední
  Náročnost: Nízká
  Kategorie: Použitelnost

2.5 Případy užití

Pro návrh systému SOS autor zvolil metodu případů užití (anglicky *Use case*), která je standardem při návrhu obdobných informačních systémů. Modelování případů užití navazuje na analýzu požadavků – jednotlivé funkční požadavky se typicky rozpadají na několik případů užití. Model případů užití slouží například jako podklad k návrhu akceptačních testů, zpřesnění odhadů pracnosti a nakonec jako zadání pro programátora, který bude systém vyvíjet.

Model případů užití sestává ze seznamu účastníků (neboli aktérů), který zároveň je znázorněn na obrázku 2.1, diagramu případů užití na obrázku 2.2 a seznamu případů užití, kde jsou jednotlivé případy definovány textovým popisem či scénářem. Součástí modelu může být i tabulka pokrytí funkčních požadavků. [10, kap. 6]

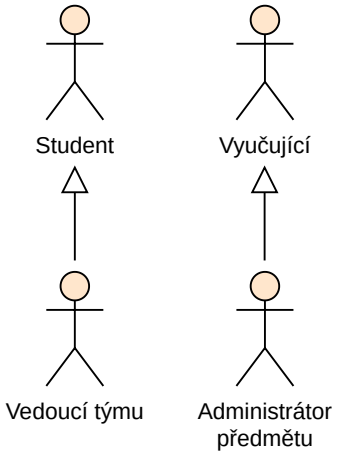
2.5.1 Seznam aktérů

Jednotlivé akce zmiňované ve funkčních požadavcích a případech užití mohou být prováděny jen některými typy uživatelů. Systém rozlišuje celkem čtyři typy uživatelů popsané v seznamu níže. Je zde nutno zmínit, že dané role uživateli v systému nepřísluší globálně, ale vždy se vztahují ke konkrétnímu předmětu v konkrétním semestru. Ten samý uživatel může být například studentem jednoho předmětu v rámci svého magisterského studia a zároveň vyučujícím některého z bakalářských předmětů. Při dostatečně dlouhém provozu systému může nastat i situace, kdy by byl uživatel studentem určitého předmětu v jednom semestru a následně vyučujícím téhož předmětu o několik semestrů později.

- **Student** – Uživatel představující studenta zvoleného předmětu. V závislosti na konfiguraci předmětu může vytvářet a upravovat zadání projektů a vytvářet týmy. V případě, že je členem nějakého týmu, může odevzdávat vypracovaná řešení.

- **Vedoucí týmu** – Muže provádět tytéž akce jako Student. Nad to může (v závislosti na konfiguraci předmětu) zvát nové členy do svého týmu nebo přijímat žádosti o přijetí do týmu. Vedení týmu může předat některému ze studentů, který je členem jeho týmu, čímž se z něj stane Student a z daného studenta Vedoucí týmu.

- **Vyučující** – Uživatel představující vyučujícího zvoleného předmětu. V závislosti na konfiguraci předmětu může vytvářet a upravovat zadání projektů a vytvářet a spravovat týmy. Vyučující také ohodnocuje vypracovaná řešení odevzdaná studenty. Může zobrazit celkový přehled bodových zisků v daném předěmtu. V závislosti na konfiguraci předmětu může tuto konfiguraci měnit pro paralelky, které vyučuje.



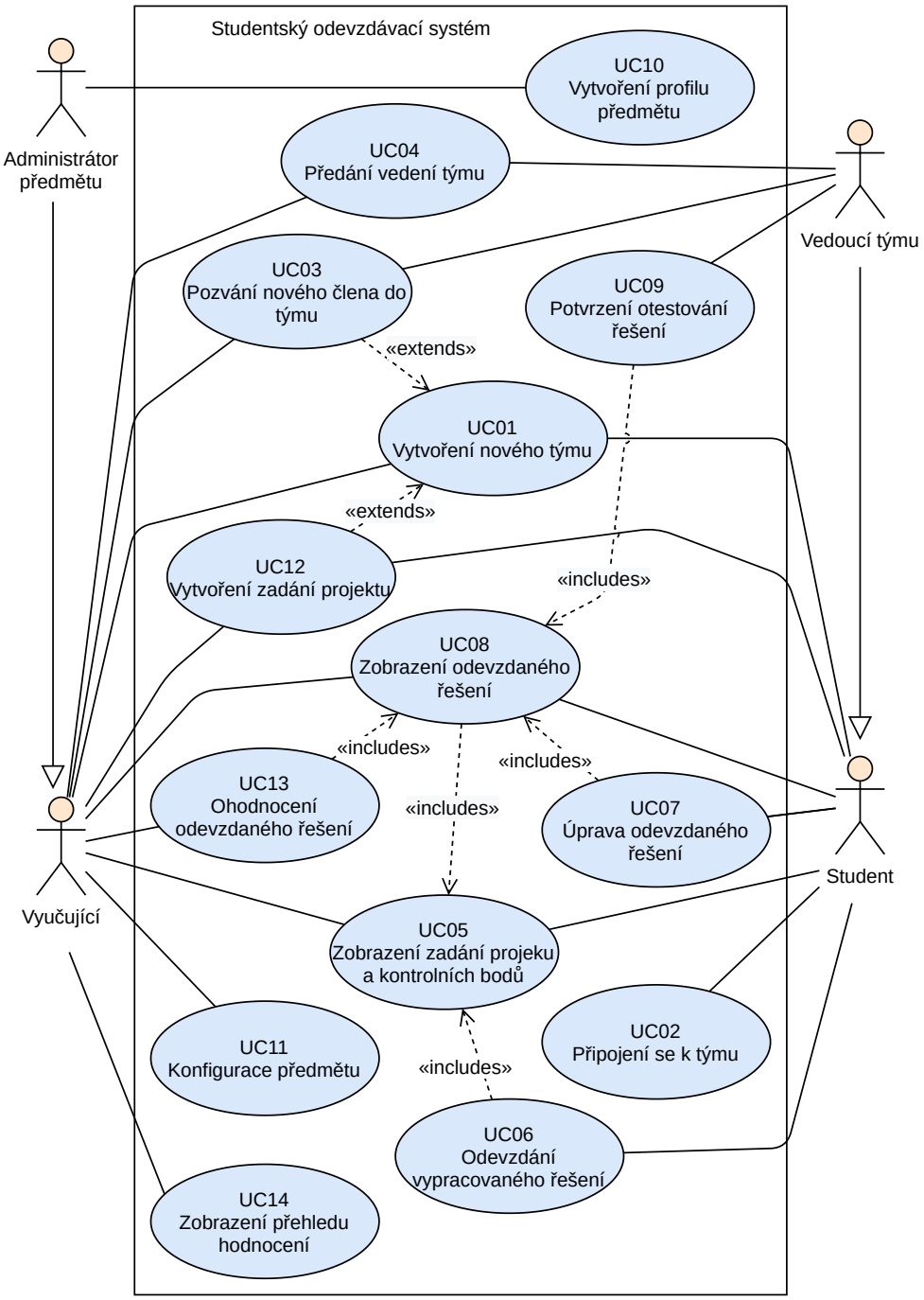
Obrázek 2.1: Diagram aktérů v modelu případů užití

- **Administrátor předmětu** – Může provádět tytéž akce jako Vyučující. Navíc může měnit globální konfiguraci předmětu, včetně povolení vytváření vlastních konfigurací jednotlivými vyučujícími. Může ostatní vyučující daného předmětu povýšit Administrátory předmětu, nebo ostatním Administrátorům předmětu tuto roli odebrat, čímž se z nich stanou Vyučující. Administrátorem předmětu se uživatel stává, jestliže je při vytváření profilu předmětu veden v systému KOS jako jeho garant. Profil předmětu může vytvořit uživatel, který je v systému KOS veden jako jeho garant.

2.5.2 Seznam případů užití

- **UC01 – Vytvoření nového týmu**: Uživatel na stránce předmětu klikne na tlačítko „Vytvořit nový tým“. Systém zobrazí formulář tvorby nového týmu. Uživatel zde buďto zvolí některé z témat zadaných vyučujícím, nebo definuje vlastní (možnosti závisí na konfiguraci předmětu). Ze seznamu studentů zvolí ty, kteří se mají stát členy týmu. Následně klikne na tlačítko „Vytvořit tým“. Systém vytvoří nový tým a rozešle vybraným studentům pozvánky k připojení se k týmu, nebo je do týmu přidá okamžitě, provádí-li akci vyučující.
  Aktér: Student nebo Vyučující

- **UC02 – Připojení se k týmu**: Student na stránce již existujícího týmu klikne na tlačítko „Připojit se k týmu“. Systém odešle vedoucímu týmu žádost studenta o připojení se k týmu. Vedoucí týmu či vyučující žádost buďto zamítne, nebo ji potvrdí, čímž se daný student stane členem týmu.



Obrázek 2.2: Diagram případů užití

Aktér: Student

- **UC03 – Pozvání nového člena do týmu**: Vedoucí týmu nebo vyučující na stránce týmu klikne na tlačítko „Spravovat tým“. Systém zobrazí seznam aktuálních členů týmu a formulář, ve kterém uživatel vybere studenta, který se má stát novým členem. Svou volbu potvrdí kliknutím na tlačítko „Pozvat nového člena“ nebo „Přidat nového člena“, pokud akci provádí Vyučující. Systém vybranému studentovi odešle pozvánku k připojení se k týmu. Daný student ji buďto zamítne nebo potvrdí, čímž se stává členem týmu. V případě, že akci provádí Vyučující, systém neodešle pozvánku, ale studenta do týmu přidá okamžitě.
  Aktér: Vedoucí týmu nebo Vyučující

- **UC04 – Předání vedení týmu**: Vedoucí týmu nebo vyučující na stránce týmu klikne na tlačítko „Spravovat tým“. Systém zobrazí seznam aktuálních členů týmu. Vedoucí týmu nebo vyučující zvolí člena týmu, který se má stát novým vedoucím a klikne u jeho jména na tlačítko „Předat vedení“, čímž se vybraný student stává Vedoucím týmu a původní vedoucí Studentem.
  Aktér: Vedoucí týmu nebo Vyučující

- **UC05 – Zobrazení zadání projektu a kontrolních bodů**: Uživatel přejde na stránku týmu. Zde mu systém zobrazí kompletní zadání, které má tým vypracovat, jednotlivé kontrolní body a jejich požadavky, včetně termínů odevzdání a maximálního a minimálního počtu získaných bodů.
  Aktér: Student nebo Vyučující

- **UC06 – Odevzdání vypracovaného řešení**: Student přejde na stránku svého týmu. U popisu kontrolního bodu ke kterému chce odevzdávat své vypracované řešení klikne na tlačítko „Nové odevzdání“. Systém zobrazí formulář odevzdání řešení. Zde má student možnost nahrát několik souborů či webových linků, kde se nachází jeho vypracované řešení. Také může napsat textovou poznámku pro vyučujícího. Následně může tlačítkem odevzdání uložit, čímž se zpřístupní pro zobrazení a úpravy ostatním členům týmu, nebo rovnou odevzdat k hodnocení vyučujícímu.
  Aktér: Student

- **UC07 – Úprava odevzdaného řešení**: Student na stránce týmu pod daným kontrolním bodem vybere jedno z odevzdání, která dosud nebyla vyučujícím ohodnocena a klikne na tlačítko „Zobrazit“. Systém studenta

přesměruje na stránku zvoleného odevzdání. Student klikne na tlačítko „Upravit odevzdání“. Systém zobrazí stejný formulář jako při vytváření odevzdání. Student může změnit poznámku pro vyučujícího a odstranit nahrané soubory a odkazy nebo přidat nové. Následně své úpravy uloží kliknutím na tlačítko „Uložit změny“. Pokud tak ještě nebylo učiněno, může odevzdat řešení k hodnocení vyučujícímu. V opačném případě může odevzdání odvolat.
Aktér: Student

- **UC08 – Zobrazení odevzdaného řešení**: Uživatel na stránce týmu pod daným kontrolním bodem vybere jedno z odevzdání a klikne na tlačítko „Zobrazit“. Systém zobrazí odevzdané řešení spolu s textovou poznámkou vyučujícího a bodovým ohodnocením, pojud již bylo odevzdání vyučujícím ohodnoceno.
Aktér: Student nebo Vyučující

- **UC09 – Potvrzení otestování řešení**: Vedoucí týmu na stránce týmu pod daným kontrolním bodem vybere jedno z odevzdání. Systém zobrazí seznam studentů, kteří se přihlásili k testování daného projektu, rozdělený na 2 části – potvrzení a nepotvrzení. Vedoucí týmu klikne na tlačítko „Potvrdit otestování“ u některého z nepotvrzených přihlášených studentů, čímž potvrdí, že daný student odevzdané řešení otestoval.
Aktér: Vedoucí týmu

- **UC10 – Vytvoření profilu předmětu**: Vyučující na domovské stránce klikne na tlačítko „Nový předmět“. Systém na základě dat získaných ze systému KOS zobrazí nabídku předmětů, jichž je daný vyučující garantem a které dosud nemají v systému vytvořený profil pro zvolený semestr. Vyučující zvolí předmět a potvrdí svou volbu kliknutím na tlačítko. Systém stáhne ze ze systému KOS potřebná data o předmětu, včetně informací o jeho vyučujících a studentech. Následně systém vytvoří profil předmětu a uživatelské účty studentů a vyučujících, kteří jsou k danému předmětu přiřazeni. Pokud již tito uživatelé mají v systému vytvořené účty, nejsou již vytvářeny znovu, ale pouze se přiřadí k nově vzniklému předmětu.
Aktér: (budoucí) Administrátor předmětu

- **UC11 – Konfigurace předmětu**: Vyučující na stránce předmětu klikne na tlačítko „Nastavení“. Systém zobrazí formulář s možnostmi konfigurace předmětu a jednotlivých kontrolních bodů. Je-li uživatel Administrátor předmětu tohoto předmětu, má zde možnost přejít na konfiguraci

výchozích nastavení předmětu. Zde také může zvolit, zda si každý Vyučující daný předmět může konfigurovát sám, nebo má být pro všechny použita konfigurace definovaná Administrátorem předmětu. V druhém případě je ostatním vyučujícím na stránce konfigurace předmětu pouze zobrazena výchozí konfigurace definovaná garantem bez možnosti provádět změny.
Aktér: Vyučující

- **UC12 – Vytvoření zadání projektu**: Vyučující zahájí tvorbu zadání kliknutím na tlačítko „Vytvořit zadání". Student je systémem vyzván k vytvoření zadání během tvorby týmu. Systém zobrazí formulář tvorby zadání. Uživatel vyplní název a popis zadání a uloží jej kliknutím na tlačítko. Později může uživatel zadání upravovat pomocí stejného formuláře.
Aktér: Student nebo Vyučující

- **UC13 – Ohodnocení odevzdaného řešení**: Vyučující přejde na stránku odevzdaného řešení buďto přímo ze stránky předmětu, kde mu systém zobrazuje odevzdání, která čekají na jeho ohodnocení, nebo ze stránky konkrétního týmu. Zde může zobrazit soubory a webové odkazy nahrané studenty a textovou poznámku studentů. Vyučující klikne na tlačítko „Upravit hodnocení". Systém zobrazí formulář hodnocení odevzdání. Zde vyučující může zadat textovou poznámku pro studenty, soukromou textovou poznámku pro sebe a počet získaných bodů, případně i penalizaci za pozdní odevzdání. Své hodnocení může uložit tlačítkem „Uložit soukromě", nebo zveřejnit pro studenty tlačítkem „Uložit hodnocení".
Aktér: Vyučující

- **UC14 – Zobrazení přehledu hodnocení**: Vyučující na stránce předmětu klikne na tlačítko „Klasifikace". Systém zobrazí tabulku obsahující jednotlivé studenty předmětu, jejich bodové zisky v jednotlivých kontrolních bodech a celkový počet získaných bodů. Vyučující může tabulku seřadit podle jednotlivých sloupců.
Aktér: Vyučující

2.6 Návrh architektury systému

2.6.1 Webová aplikace

Z výše uvedených nefunkčních požadavků jasně vyplývá, že se nutně musí jednat o webovou aplikaci. Webová aplikace je aplikace poskytovaná z webového

	FP1	FP2	FP3	FP4	FP5	FP6	FP7
UC01				+			
UC02				+			
UC03				+			
UC04				+			
UC05			+				
UC06					+		
UC07					+		
UC08					+		
UC09						+	
UC10	+	+					
UC11	+						
UC12			+				
UC13							+
UC14							+

Tabulka 2.1: Tabulka pokrytí funkčních požadavků

serveru prostřednictvím počítačové sítě. O zobrazení webové aplikace na cílovém zařízení se stará webový prohlížeč, není vyžadována instalace na zařízení uživatele. Aktualizace taktéž probíhají na serveru, uživatelé mají tedy vždy k dispozici nejnovější verzi aplikace bez nutnosti instalace aktualizací. Další výhodou je možnost přístupu k aplikaci z jakéhokoliv zařízení, které disponuje webovým prohlížečem. Nevýhodou webové aplikace je především závislost na internetovém připojení. [11]

2.6.1.1 Jednostránková aplikace

Jednostránková aplikace, neboli SPA (z anglického *Single-page application*) je typ webové aplikace, která část aplikační logiky přenáší na stranu klienta. Server se zajišťuje pouze práci s daty, ke kterým klientská část aplikace přistupuje například prostřednictvím REST API. Klientská část aplikace je realizována jedinou webovou stránkou, jejíž struktura je na základě určitých událostí (především interakcí s uživatelem) průběžně aktualizována. To zajišťuje skriptovací jazyk běžící ve webovém prohlížeči, například JavaScript.

Výhodou jednostránkové aplikace je především mnohonásobně rychlejší odezva na uživatelské požadavky, jelikož buďto vůbec nedochází ke komunikaci se serverem, nebo jsou přenášena pouze potřebná data a nikoliv celé webové stránky. Tento přístup nicméně vyžaduje stažení vešekerého kódu klienstké části aplikace najednou při prvním načtení aplikace, které tak může trvat řádově déle než je tomu u vícestránkové aplikace. Nevýhodou jednostránkové aplikace je její naprostá závislost na skriptovacím jazyku běžícím ve webovém prohlížeči. Některé prohlížeče mohou mít tuto funkcionalitu vy-

pnutou, v důsledku čehož nemůže jednostránková aplikace správně fungovat. [12, kap. 3]

2.6.1.2 Vícestránková aplikace

Vícestránková aplikace, neboli MPA (z anglického *Multi-page application*) je typ webové aplikace, u které se veškerá aplikační logika nachází na straně serveru. Server a klient spolu komunikují prostředníctvím protokolu HTTP. Dynamické chování vícestránkové aplikace je zajištěno tím, že server reaguje na vstupy uživatele, jako je odeslání vyplněného formuláře či přechod na jinou URL adresu. Tyto vstupy jsou serveru odesílány v podobě HTTP požadavků. Server v reakci na tyto vstupy vždy vygeneruje kompletní webovou stránku s požadovaným obsahem, kterou následně odešle klientovi, který ji již pouze zobrazí.

Vícestránkové aplikace jsou vhodnější pro obsah, který je přirozeně rozvětvený do více kategorií a dává tedy větší smysl jej strukturovat do více různých stránek. Výhodou vícestránkové aplikace je ve většině případů schopnost správně fungovat i bez potřeby využití skriptovacího jazyka. Nevýhodou je již zmíněná potřeba v reakci na každý uživatelský požadavek načítat novou webovou stránku, což může trvat řádově i několik sekund, především v případě nekvalitního internetového připojení či přetížení serveru. Další nevýhodou vícestránkové aplikace je pevnější svázání backendové a frontendové části aplikace než u jednostránkové aplikace. [12, kap. 2]

2.6.1.3 Využití pro systém SOS

Autor se nakonec po uvážení výše zmíněných výhod a nevýhod rozhodl pro implementaci systému SOS jako vícestránkové aplikace. Jedním z hlavních argumentů je, že uživatelské rozhraní systému SOS lze přirozeně strukturovat do několika částí, které spolu nejsou příliš propojeny a dává větší smysl, aby se nacházely na samostatných stránkách. Konkrétně se jedná například o domovskou stránku s přehledem všech předmětů, stránku konkrétního předmětu se seznamem projektů, stránky jednotlivých týmů/projektů, nebo různé sekce dostupné pouze vyučujícím, jako je přehled klasifikace či konfigurace předmětu. Nicméně u některých sekcí aplikace, jako je například správa členů týmu či odevzdávání vypracovaných řešení, by byla možnost dynamické aktualizace obsahu daná jednostránkovým přístupem užitečná.

Dalším neopomenutelným faktorem při rozhodování je to, že framework Django, který se autor pro implementaci systému rozhodl využít a detailněji se o něm zmiňuje dále v této práci, je využitelný především pro tvorbu vícestránkových aplikací. Lze jej sice využívat i pouze jako backend, který poskytuje přístup k datům pomocí REST API, nicméně by tento přístup vyžadoval využití některého dalšího frameworku pro implementaci klientské části aplikace. Tuto cestu autor práce shledal jako nevhodnou také proto, že s frontendovými

frameworky, jako jsou například React či Angular (který využívá například již existující systém NURIS) a vůbec se samotným jazykem JavaScript, který by v takovém případě bylo nutné masivně využívat, nemá příliš velké zkušenosti.

2.6.2 Architektonický vzor

Při vývoji rozsáhlejších aplikací je vhodné využít některý ze známých architektonických vzorů. Architektonický vzory vyjadřuje základní strukturální organizaci nebo schéma softwarového systému. Poskytuje množinu definovaných podsystémů, specifikuje jejich zodpovědnosti a obsahuje pravidla pro organizaci vztahů mezi nimi. Tento přístup zpřehledňuje kód a tím usnadňuje vývoj a podporuje dlouhodobou udržitelnost projektu. [13, kap. 2]

2.6.2.1 Model-View-Controller

MVC neboli Model-View-Controller patří mezi nejpopulářnější architektonické vzory používané při vývoji webových aplikací. Tato architektura dělí aplikaci na tři logické částí – Model, View a Controller.

Model zprostředkovává práci s daty a je zodpovědný za jejich konzistenci. Nejedná se pouze o sadu datových objektů, kromě reprezentace dat je součástí modelu je součástí Modelu i veškerá aplikační logika.

View představuje uživatelské rozhraní aplikace. U webových aplikací se jedná o kód, který se s využitím dat z Modelu stará o generování odpovědi, která bude nakonec odeslána klientovi. Výstupem je typicky HTML dokument, ale často se také jedná například o reprezentaci dat ve formátu JSON či XML.

Controller propojuje jednotlivé komponenty aplikace. Reaguje na přicházející vstupy ve formě HTTP požadavků, vyvolává příslušné změny v Modelu a obstarává z něj data, která poté předává View ke zobrazení. [14]

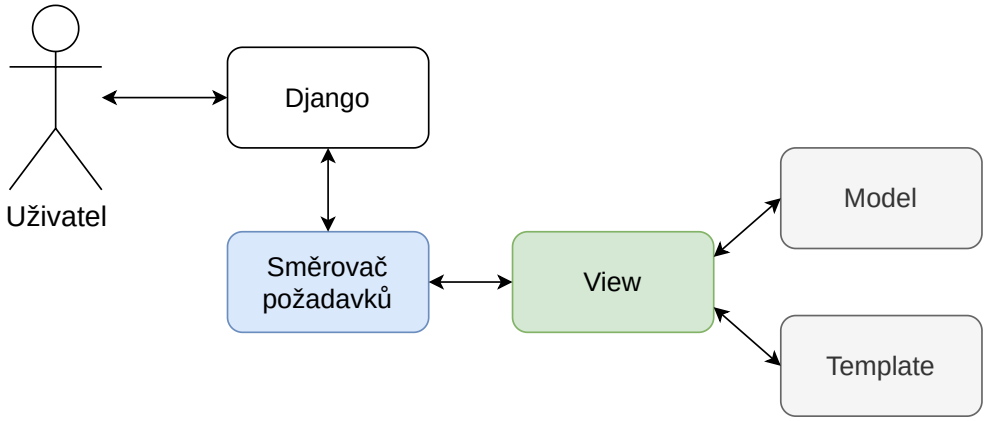
2.6.2.2 Model-View-Template

Architektonický vzor MVT neboli Model-View-Template je jednou z mnoha modifikací vzoru MVC. Opět rozděluje aplikaci na tři logické částí - tentokrát Model, View a Template.

Model podobně jako v MVC zodpovídá práci s daty. Rozdíl je v tom, že v MVT model nemusí obsahovat veškerou aplikační logiku, ta je částečně přenesena na View.

Template je komponenta, která reprezentuje uživatelské rozhraní aplikace. Narozdíl od View v MVC se však jedná pouze o textové šablony, které slouží ke generování odpovědí, ať už HTML dokumentů nebo jiných formátů.

Komponenta View plní v MVT částečně podobnou funkci, jako Controller u MVC. Reaguje na příchozí požadavky a obstarává data z Modelu. Narozdíl od MVC se zde ale také nachází část aplikační logiky. Architektura MVT přesně nestanovuje, podle jakých pravidel má být aplikační logika umísťována do Modelu nebo do View, záleží tak na konkrétním případě a na rozhodnutí



Obrázek 2.3: Diagram návrhového vzoru MVT

vývojáře. Získaná a případně modifikovaná data z Modelu následně View vloží do příslušné šablony a vygeneruje odpověď, která bude odeslána klientovi. [15, kap. 5.2]

Architektonický vzor MVT je modifikací vzoru MVC specifickou pro framework Django, který se autor práce rozhodl využít. Z využití tohoto frameworku vyplývá i nutnost použití architektonického vzoru MVT.

2.7 Uživatelské rozhraní

2.7.1 Responzivní design

Vzhledem k tomu, že se jedná o webovou aplikaci, je potřeba řešit její správné zobrazení a fungovaní na všech typech a velikostech zařízení. Uživatelé mohou prostřednictvím webového prohlížeče aplikaci využívat jak na klasických stolních počítačích disponujících obrazovkami s velkým rozlišením, tak i na mobilních telefonech či tabletech. Podle aktuálních dat využívá webové stránky a aplikace v průměru více než polovina uživatelů právě z mobilních zařízení. [16] Zajištění správného fungování aplikace nezávisle na velikosti obrazovky cílového zařízení se tedy v dnešní době jeví jako velmi potřebné k dosažení její dobré použitelnosti všemi jejími uživateli.

Z této skutečnosti jasně plyne požadavek na responzivní design této webové aplikace. Responzivní design je pojem, jenž popisuje přístup k návrhu webových stránek a rozhraní webových aplikací, jehož hlavní myšlenkou je právě zohlednění typu a velikosti zařízení využívaného pro zobrazování obsahu. [17]

Při realizaci responzivního designu lze obecně postupovat dvěma způsoby – *mobile-first* a *desktop-first*, což znamená se primárně zaměřit na malou obrazovku mobilního zařízení, resp. na velkou obrazovku stolního počítače. V současnosti je bývá z důvodu stále většího zastoupení mobilních zařízení upřednostňován *mobile-first* přístup, což potvrzuje například i doporučení společ-

nosti Google, která nedávno zavedla tzv. *mobile-first indexing* [18], což v praxi znamená, že se jejich internetový vyhledávač bude řídit mobilními verzemi webů.

Systém SOS nicméně nemá být komerční web, u kterého je potřeba se zabývat optimalizací pro internetové vyhledávače. Navíc autor očekává, že systém bude primárně využíván ze stolního počítače či laptopu, jelikož tam studenti vytvářejí svá řešení, která budou do systému odevzdávat. Mobilní verzi pak budou využívat spíše pro zobrazování hodnocení či správu týmů. Vyučující budou také pravděpodobně častěji do systému přistupovat z počítače, protože zobrazování odevzdaných řešení studentů na mobilních zařízeních by mohlo být nepohodlné nebo až nemožné, například v případě zdrojových kódů, které je potřeba stáhnout a spustit. Z tohoto důvodu bylo uživatelské rozhraní navrhováno s přístupem *desktop-first*, což je i důvodem, proč drátěný model nacházející se dále v této sekci modeluje uživatelské rozhraní pro velkou obrazovku.

2.7.2 Návrh uživatelského rozhraní

Základní strukturu uživatelského rozhraní autor navrhl pomocí tzv. drátěného modelu neboli *wireframe*. Jedná se jednoduché schéma, které definuje strukturu a funkci jednotlivých obrazovek budoucí aplikace [19, kap. 10]. Pro tvorbu těchto schémat autor použil aplikaci Balsamiq[1].

Návrh obsahuje popis funkcí a struktury jednotlivých obrazovek, včetně vazby na kokrétní případy užití. Příslušné drátěné modely jednotlivých obrazovek se pak nachází v příloze B. Součástí návrhu jsou i mapy obrazovek, které znázorňují přechody mezi nimi.

Návrh uživatelského rozhraní není příliš detilní, jednotlivé obrazovky neobsahují všechny informace, které bude finální uživatelské rozhraní zobrazovat a rozmístění jednotlivých prvků je pouze orientační. Také nepokrývá všechny případy užití, konkrétně UC09 a UC14 návrh vůbec neřeší. Po vytvoření tohoto stručného návrhu se autor rozhodl, že o provedení zbylých funkcionalit rozhodne až během vývoje.

Jedním z důvodů pro toto rozhodnutí byla skutečnost, že je potřeba uživatelské rozhraní realizovat tak, aby bylo dobře použitelné na všech velikostech obrazovek. Detailní a přesný návrh by tedy musel obsahovat nejméně tři varianty každé obrazovky s jiným rozmístěním prvků – pro mobilní telefon, tablet a velký počítačový monitor – což by vedlo k nepřiměřené časové náročnosti.

Také bylo pro autora obtížné na základě drátěných modelů vyhodnotit, jestli bude výsledné uživatelské rozhraní dostatečně přehledné a uživatelsky přívětivé. Autor tedy usoudil, že lepší představu získá až během vývoje, kdy uvidí reálnou podobu vznikajícího uživatelského rozhraní. Dalším důvodem pro vytvoření pouze takto stručného drátěného modelu bylo to, že před zahá-

[1] `https://balsamiq.com/wireframes/`

jením vývoje systému nebyl ještě kompletně rozmyšlen způsob realizace všech požadovaných funkcionalit z hlediska aplikační logiky a uživatelského workflow, což představovalo riziko, že by detailní návrh všech obrazovek nebylo možné jedoduše využít.

Finální podoba uživatelského rozhraní a zejména její odlišnosti od tohoto návrhu jsou popsány a odůvodněny v následující kapitole, která se věnuje realizaci systému.

2.7.2.1 Společné prvky

Součástí všech stránek aplikace je postranní panel s přehledem studovaných předmětů, které zároveň slouží jako odkazy na příslušné stránky. Pod nimi se nachází tlačítka pro výběr semestru. Dále je součástí všech obrazovek horní lišta s odkazem na domovskou stránku, přehledem upozornění na důležité události v systému, volbou jazyka uživatelského rozhraní a tlačítkem pro odhlášení se z aplikace.

2.7.2.2 Domovská stránka – student

Na stránce obrazovce je studentovi zobrazen přehled předmětů pro vybraný semestr. Kliknutím na předmět student přejde na stránku tohoto předmětu. Dále se zde nachází přehled všech projektů, na kterých student pracuje, na jejichž detail lze rovněž přejít kliknutím na ně.
Wireframe: Obrázek B.1

2.7.2.3 Předmět – student

Na stránce předmětu je studentovi zobrazen projekt, na kterém pracuje. Pokud student zatím na žádném projektu nepracuje, je zde místo projektu tlačítko „Vytvořit nový projekt“. Toto tlačítko je zobrazeno pouze v případě, že konfigurace předmětu povoluje správu zadání projektů studenty. Dále je součástí této stránky přehled ostatních projektů v rámci vybraného předmětu. Pokud mají tyto projekty a uživatel volnou kapacitu, nachází se u projektů tlačítka pro připojení se k projektu jako člen týmu nebo tester.
Řešené případy užití: UC02
Wireframe: Obrázek B.2

2.7.2.4 Založení a úprava projektu – student

Na této stránce je studentovi zobrazen formulář pro vytvoření nového projektu. Stejný formulář lze následně vytvořit k úpravám již vytvořeného projektu, v takovém případě se zobrazí předvyplněný existujícími hodnotami a místo tlačítka „Vytvořit projekt“ bude tlačítko „Uložit“.

Řešené případy užití: UC01
Wireframe: Obrázek B.3

2.7.2.5 Projekt/tým – student

Na stránce projektu je studentovi zobrazeno zadání, které tým vypracovává a přehled jednotlivých kontrolních bodů s informacemi o termínech odevzdání a současném stavu, případně počet získaných bodů. Dále je na této stránce přehled členů týmu a testerů. Pokud je student vedoucím týmu a konfigurace předmětu to dovoluje, jsou zde k dispozici tlačítka, která studenta přesměrují na stránku úprav zadání a stránku správy týmu. Pokud student není členem žádného týmu a konfigurace předmětu to dovoluje, zobrazí se na stránce tlačítko „Připojit se k týmu“, kterým je vedoucímu daného týmu nebo vyučujícímu odeslána žádost o přijetí studenta do tohoto týmu.
Řešené případy užití: UC02, UC05
Wireframe: Obrázek B.4

2.7.2.6 Správa týmu – student

Na stránce správy týmu je studentovi zobrazen kompletní přehled všech členů týmu, testerů a zodpovědý vyučující. Kliknutím na tlačítko „Zpráva“ u člena týmu je student přesměrován do svého výchozího poštovního klienta, kde bude předvyplněna fakultní e-mailová adresa vybraného člena týmu. Pokud je student vedoucím týmu, zobrazí se dále u členů týmu tlačítko „Předat vedení“, kterým může vedení týmu předat jinému členovi týmu. Pokud není kapacita týmu plná a student je vedoucí týmu, zobrazí se na této stránce formulář pro pozvání dalších studentů do týmu.
Řešené případy užití: UC03, UC04
Wireframe: Obrázek B.5

2.7.2.7 Kontrolní bod – student

Na stránce kontrolního bodu je studentovi zobrazen popis požadavků na odevzdání daného kontrolního bodu, termín odevzdání a možný počet získaných bodů. Dále se zde nachází seznam již odevzdaných řešení k tomuto kontrolních bodů spolu s informací o stavu těchto odevzdání. Kliknutím na odevzdané řešení student může zobrazit detaily v podobě odevzdaných souborů, textové poznámky studentů a textové poznámky vyučujícího. Možné stavy jsou *Čeká na zhotovení*, *Čeká na ohodnocení*, *Vráceno k přepracování* a *Schváleno*. Neexistuje-li zatím k danému kontrolnímu bodu žádné odevzdané řešení schválené vyučujícím, je zde k dispozici formulář pro odevzdání řešení, který umožňuje nahrávat soubory, vložit textovou poznámku a potvrdit odevzdání.

Řešené případy užití: UC05, UC06, UC07, UC08
Wireframe: Obrázek B.6

2.7.2.8 Kontrolní bod s hodnocením – student

Jedná se o stejnou stránku jako v předešlém bodě, pouze s tím rozdílem, že odevzdané řešení k danému kontrolnímu bodu již bylo schváleno vyučujícím. Je zde tedy zobrazena informace o tom, že odevzdané řešení bylo schváleno a získaný počet bodů. Na schématu je zároveň znázorněno zobrazení informací o odevzdaném řešení zmiňované v minulém bodě.
Řešené případy užití: UC05, UC06, UC07, UC08
Wireframe: Obrázek B.7

2.7.2.9 Domovská stránka – vyučující

Domovská stránka vyučujícího obsahuje přehle vyučovaných předmětů, na jejichž stránky lze přejít kliknutím. Dále je zde vyučujícímu zobrazen přehled odevzdaných řešení studentů, která čekají na jeho ohodnocení, na která může rovněž přejít kliknutím. V horní části stránky je k dispozici tlačítko, které zobrazí formulář pro vytvoření nového předmětu v systému SOS. Vyučující dostane na výběr z předmětů, které zatím v systému SOS nejsou a které podle dat ze systému KOS garantuje. Po výběru předmětu je předmět vytvořen a vyučující přesměrován na konfiguraci tohoto předmětu.
Řešené případy užití: UC10
Wireframe: Obrázek B.8

2.7.2.10 Předmět – vyučující

Na stránce předmětu systém vyučujícímu zobrazí seznam zadání, která v rámci daného předmětu vytvořil (je-li vytváření zadání vyučujícími dáno konfigurací předmětu) a seznam paralelek, které daný vyučující učí. V horní části stránky jsou k dispozici tlačítka, která vyučujícího přesměrují na stránku konfigurace předmětu, seznam kontrolních bodů nebo formulář tvorby nového zadání (opět v závislosti na konfiguraci).
Wireframe: Obrázek B.9

2.7.2.11 Konfigurace předmětu – vyučující

Na stránce konfigurace předmětu má vyučující k dispozici formulář pro nastavní všech hodnot konfigurace vybraného předmětu. Na tuto stránku má přístup pouze vyučující s administrátorskými oprávněními pro daný předmět.

Řešené případy užití: UC11
Wireframe: Obrázek B.10

2.7.2.12 Zadání – vyučující

Stránka tvorby nebo úpravy nového zadání z pohledu vyučujícího je stejná jako stránka založení nového projektu z pohledu studenta, liší se pouze v úvodním textu. Zobrazený formulář lze opět později využít i k úpravě již vytvořeného zadání.

Řešené případy užití: UC12
Wireframe: Obrázek B.11

2.7.2.13 Paralelka – vyučující

Na stránce paralelky je vyučujícímu zobrazen seznam odevzdaných řešení studentů dané paralelky, která čekají na jeho ohodnocení. Pod ním se nachází seznam ostatních projektů v rámci dané paralelky. Kliknutím na položku v seznamu je vyučující přesměrován na stránku příslušného projektu.
Wireframe: Obrázek B.12

2.7.2.14 Seznam kontrolních bodů – vyučující

Na této stránce je vyučujícímu zobrazen seznam jednotlivých kontrolních bodů včetně jejich termínů odevzdání. V případě, že to konfigurace předmětu dovoluje, se zde také nachází tlačítko „Přidat kontrolní bod“, které vyučujícího přesměruje na formulář tvorby nového kontrolního bodu. Kliknutím na jeden z existujících kontrolních bodů je vyučující přesměrován buďto na formulář úpravy již existujícího kontrolního bodu, nebo pouze na stránku s popisem tohoto kontrolního bodu – to rovněž závisí na konfiguraci předmětu.
Řešené případy užití: UC05, UC11
Wireframe: Obrázek B.13

2.7.2.15 Tvorba/úprava kontrolního bodu – vyučující

Na této stránce je vyučujícímu zobrazen formulář pro vytvoření nového kontrolního bodu či úpravu již existujícího. Do formuláře vyučující zadá název kontrolního bodu, textový popis požadavků, termín odevzdání a maximální počet bodů, které mohou studenti za odevzdání řešení získat.
Řešené případy užití: UC05, UC11
Wireframe: Obrázek B.14

2.7.2.16 Projekt/tým – vyučující

Stránka konkrétního projektu je z pohledu vyučujícího stejná, jako z pohledu studenta. Pouze opět závisí na konfiguraci, zda-li se zde budou nacházet tlačítka pro přechod na úpravu vypracovávaného zadání a správu týmu. Stejně jako student i vyučující kliknutím na konkrétní kontrolní bod v přehledu přejde na detail tohoto kontrolního bodu.
Řešené případy užití: UC05
Wireframe: Obrázek B.15

2.7.2.17 Kontrolní bod projektu – vyučující

Stránka kontrolního bodu v rámci konkrétího projektu vyučujícímu podobně jako z studentovi zobrazuje zadání tohoto kontrolního bodu a přehled již odevzdaných řešení. Kliknutím na může vyučující zobrazit detaily konkrétního odevzdání. V dolní částí je zobrazeno poslední odevzané řešení a formulář pro ohodnocení tohoto řešení. Formulář umožňuje vyučujícímu zadat textovou poznámku a počet získaných bodů.
Řešené případy užití: UC05, UC08, UC13
Wireframe: Obrázek B.16

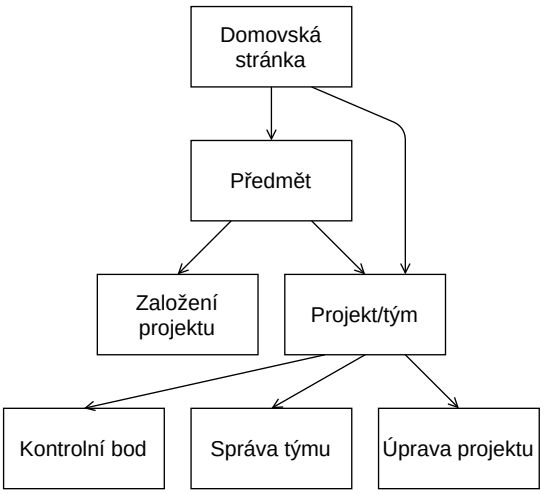
2.7.2.18 Mapy obrazovek

Návrh obsahuje dvě mapy obrazovek, která každá popisují jedno oddělené uživatelské rozhraní – studenta a vyučujícího. Diagramy zachycují dopředné přechody mezi výše uvedenými obrazovkami. K přechodům dochází buďto kliknutím na tlačítko, jehož popisek odpovídá dané obrazovce (např. tlačítko „Nastavení předmětu“ na stránce „Předmět“ uživatele přesměruje na obrazovku „Nastavení předmětu“), nebo kliknutím na element reprezentující zobrazený objekt (kliknutím na položku v seznamu paralelek je uživatel přesměrován na stránku paralelky) – v diagramech by tedy pojmenované přechody pouze duplikovaly názvy obrazovek.

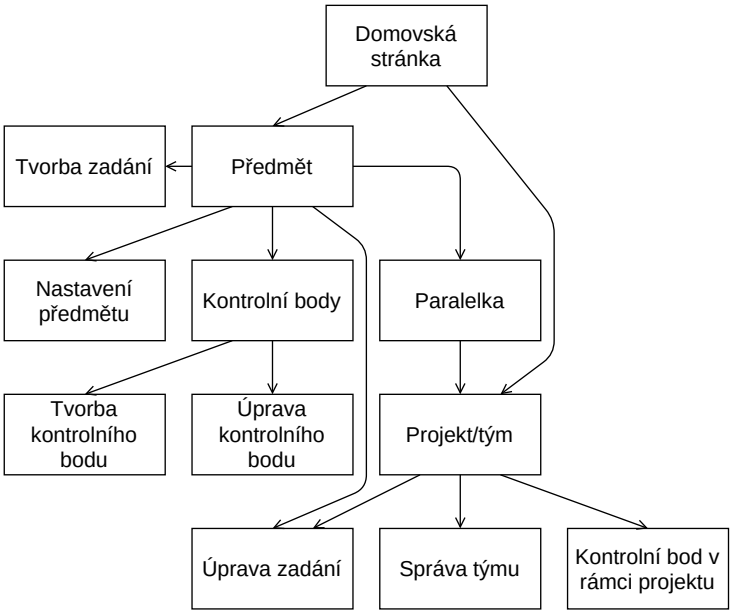
2.8 Datové úložiště

Pro realizaci perzistence dat autor zvolil relační databázi, což je dnes na trhu jednoznačně dominantní řešení [20]. Tuto volbu autor učinil především ze dvou důvodů.

Prvním z nich je skutečnost, že framework Django zvolený k implementaci systému SOS sám o sobě podporuje pouze použití relační databáze [21]. Existují sice rozšíření třetích stran, která umožňují použití například dokumentové databáze MongoDB [22], nicméně v zájmu spolehlivosti řešení se autor rozhodl taková rozšíření nepoužít.



Obrázek 2.4: Mapa obrazovek z pohledu studenta



Obrázek 2.5: Mapa obrazovek z pohledu vyučujícího

Relační databáze byla zvolena i pro systém NURIS, vzhledem k povaze tohoto systému a dat, se kterými pracuje. Analýzou možností uložení dat a jejich vhodnosti pro takový systém se zabývá diplomová práce, z níž tento systém vzešel [1, kap. 3.7]. Vzhledem k tomu, že systém SOS realizuje podobné funkcionality jako systém NURIS a dále je rozšiřuje a zobecňuje pro širší využití, vnímá autor práce tento fakt jako další argument pro využití relační databáze pro systém SOS.

2.9 Webový server

Pro provoz webové aplikace je potřeba tzv. webový server. Tímto pojmem je označován jednak hardware, tedy počítač, na kterém je systém provozován, ale také software, který zajišťuje komunikaci s klientem (internetovým prohlížečem).

2.9.1 HTTP server

Webový server ve smyslu software bývá také označován jako HTTP server. Jedná se o program, který s klientem komunikuje prostřednictvím HTTP protokolu. Když od klienta obdrží HTTP požadavek, nalezne příslušný dokument a odešle jej klientovi. V případě že jedná o statický obsah, webový server pouze odešle klientovi obsah požadovaného souboru. Zpracování dynamického obsahu, což může být například webová stránka realizovaná pomocí šablon, které mají být vyplněny daty získanými z databáze, se liší napříč implementacemi – buďto může být realizováno samotným HTTP serverem, nebo je požadavek předán tzv. aplikačnímu serveru, který obsah připraví a předá zpět HTTP serveru, který jej odešle klientovi. [23]

2.9.2 WSGI

Web Server Gateway Interface – zkráceně WSGI – je standardizované rozhraní pro komunikaci webového serveru s Python aplikacemi, definované standardem PEP 3333 [24]. Takové rozhraní redukuje provázanost webového serveru a samotné webové aplikace, čímž odstraňuje nutnost použití konkrétního webového serveru v důsledku volby aplikačního frameworku či naopak.

2.10 Ověřování totožnosti uživatelů

Vzhledem k tomu, že systém SOS bude pracovat s osobními a uživatelskými daty studentů a vyučujících, je potřeba zajistit spolehlivé ověřování totožnosti uživatelů. K tomuto účelu používají informační systémy Fakulty informačních technologií buďto technologii Shibboleth SSO [25], nebo vlastní autorizační server používající protokol OAuth 2.0 [26].

Autor práce se rozhodl pro implementaci ověřování totožnosti protokolem OAuth 2.0, především z toho důvodu, že stejný způsob využívají i systémy, se kterými musí nebo potenciálně může být systém SOS integrován. Konkrétně se jedná o KOSapi, které je nutné využívat k získávání dat o uživatelích a vyučovaných předmětech, ale potenciálně také například systémy FIT Klasifikace nebo Courses. Díky využití stejné technologie pro ověřování totožnosti uživatelů bude integrace s těmito systémy jednodušší.

Další motivací pro implementaci ověřování totožnosti protokolem OAuth 2.0 je také to, že autor práce již má s jeho implementací zkušenosti. Tento protokol je dnes poměrně běžně využíván pro přihlašování uživatelů nejen do webových, ale teké mobilních či desktopových aplikací. Autor práce se věnuje vývoji webových aplikací i ve svém zaměstnání a v nedávné době implementoval právě přihlašování uživatelů do webové aplikace prostřednictvím OAuth 2.0 autorizačních serverů poskytovanými společnostmi Google a LinkedIn.

Protokol OAuth 2.0 detailně popisuje standard RFC 6749 [27]. Základní popis fungování protokolu v českém jazyce a návod k použití fakultního autorizačního serveru je také dostupný na webu fakulty [26].

2.11 Získávání dat

Systém SOS ke svému fungování potřebuje pracovat jednak se základními daty o jednotlivých uživatelích, ale také s rozvrhovými daty. Je potřeba udržovat informace o tom, který vyučující vyučuje který předmět, kteří studenti studují který předmět, do jaké patří paralelky a podobně.

Tato data by jistě bylo možné do systému zadávat ručně, takový postup by ale byl velmi nepohodlný, zdlouhavý a náchylný k chybám. Pro takové případy všechna tato data poskytuje fakultní informační systém KOS prostřednictvím svého API [28].

Systém SOS bude získávat data o samotných uživatelích při jejich prvním přihlášení a také data o jednotlivých předmětech a rozvrzích při vytváření profilu daného předmětu v systému SOS. Způsob získání a zpracování těchto dat je popsán v následující kapitole, která se věnuje realizaci systému.

2.12 Systém pro správu verzí

Pro vývoj softwarových projektů je dnes standardem využití nějakého systému pro správu verzí. Takový systém zaznamenává veškeré změny provedené ve zdrojových kódech projektu. Umožňuje prohlížet starší verze souborů a případně se k těmto verzím vracet. Bez použití nějakého verzovacího systému by navíc byla až téměř nemožná efektivní spolupráce více vývojářů na jednom projektu. Verzovací systémy se dělí do dvou kategorií – centralizované a distribuované. [29]

2.12.1 Centralizované verzovací systémy

V centralizovaných verzovacích systémech je kompletní historie změn uložena pouze na serveru, zatímco na klientech bývá jen poslední verze a rozpracované soubory. Tyto systémy pak pracují na principu *zamknout-upravit-odemknout*. Vývojář uzamčením souboru zamezí ostatním v provádění změn v daném souboru, provede své změny, uloží je na server a následně soubor opět odemkne pro ostatní. Mezi centralizované verzovací systémy patří například CVS nebo Subversion.

2.12.2 Distribuované verzovací systémy

V případě distribuovaných (někdy také decentralizovaných) verzovacích systému nejsou role serveru a klientů pevně dané a mohou se dynamicky měnit. Kompletní historie je uložena ve všech jednotlivých uzlech, což přináší řadu výhod, jako je například možnost práce offline. Hlavní vlastností distribuovaných systémů pro správu verzí je možnost paralelní práce v lokálních větvích podle principu *zkopírovat-upravit-sloučit*. Mezi nejznámější disrtibuované verzovací systémy patří například Git a Mercurial.

2.12.3 Srovnání a využití

Pro správu historie textových souborů, jako jsou například zdrojové kódy, jsou v dnešní době využívány především distribuované verzovací systémy, jelikož umožňují efektivnější paralelní práci více vývojářů v lokálních větvích a následné slučování změn. Naopak v případě binárních souborů (například grafických) může být vhodnější využití centralizovaných verzovacích systémů, vzhledem k tomu, že se s takovými soubory musí vždy pracovat jako s celky. Paralelní provádění změn a následné slučování v takovém případě nepřichází v úvahu. Naopak je vhodné, aby byl před prováděním změn soubor uzamčen a změny v jeden okamžik prováděl pouze jeden vývojář.

2.13 Continuous Integration / Continuous Deployment

Fowler v článku z roku 2006 [30] popisuje CI neboli *Continuous Integration* jako praktiku, při níž vývojáři pravidelně integrují změny v kódu do sdíleného repozitáře. Každá verze je pak podrobena automatickým testům, čímž je zajištěno, že případné problémy budou odhaleny co nejdříve. *Continuous Integration* má největší přínos při spolupráci velkého množství vývojářů, jelikož v takovém případě bývá náročnější odhalit původ chyb, které vyplývají z nepodařené integrace změn provedených různými vývojáři.

Pojem *Continuous Deployment* bývá zaměňován s pojmem *Continuous Delivery*, skrývajícím se za stejnou zkratkou CD. *Continuous Delivery* zna-

mená stálé udržování kódu v nasaditelném stavu. *Continuous Deployment* tuto praktiku posouvá dále, jedná se o automatické nasazování takového kódu do produkce, bez dalšího lidského zásahu [31]. Jedná se tedy o praktiku navazující na *Continuous Integration*, která předpokládá, že byly změny dostatečně otestovány v testovacím prostředí automatickými testy, a tudíž by měl kód bez problémů fungovat i v produkčním prostředí. Za předpokladu, že se testovací a produkční prostředí příliš neliší tato praktika ani nepřináší příliš práce navíc, jelikož lze využít pro nasazení do produkčního prostředí podobných mechanizmů, jako u testovacího prostředí. Jak ale Fowler zmiňuje ve svém článku, je při praktikování automatického nasazování vhodné mít k dispozici také mechanismy pro snadné zvrácení nasazených změn, protože automatické testy neumusí vždy odhalit všechny potenciální problémy.

Systém SOS se potenciálně může stát velkým projektem, na jehož rozvoji by v budoucnu mohlo pracovat více vývojářů, například z řad studentů fakulty. V případě nasazení systému do provozu je také zapotřebí zajistit, že produkční verze systému bude obsahovat co nejméně chyb a že případné změny a opravy bude možné nasadit jednoduchým a spolehlivým způsobem. Z tohoto důvodu autor usoudil, že by měl i systém SOS využívat nějakou formu CI/CD. V dnešní době pro to také existuje řada nástrojů, jako jsou například CircleCI, Travis CI, či fakultou často využívaný GitLab.

2.14 Použité technologie

2.14.1 Django

Pro vývoj webové aplikace se nabízí možnost využít některý z existujících frameworků, které implementují množství funkcionalit, které by jinak bylo při vývoji každé webové aplikace programovat znovu.

Pro implementaci systému SOS autor především z důvodů svých zkušeností zvolil technologii Django[2], kterou více než rok používá ve svém zaměstnání. Nabízely se další frameworky podobného typu, jako například Ruby on Rails[3] či ASP.NET Core od společnosti Microsoft[4], u kterých ovšem autor po obeznámení se s poskytovanými funkcionalitami shledal, že nenabízejí oproti frameworku Django navíc nic, co by vyvážilo nutnost naučit se pracovat s novou technologií, se kterou autor práce nemá žádné zkušenosti.

Stejně tak byla zavržena i možnost implementovat serverovou část aplikace například pomocí minimalistického frameworku Express.js[5] postaveného na platformě Node.js[6] a klientskou část pomocí některého z populárních fron-

[2] `https://www.djangoproject.com/`
[3] `http://rubyonrails.org/`
[4] `https://docs.microsoft.com/en-us/aspnet/core/introduction-to-aspnet-core`
[5] `https://expressjs.com/`
[6] `https://nodejs.org/en/`

tendových frameworků, jako jsou React.js[7] či Vue.js[8]. Takové řešení by sice také mohlo být pro systém SOS vhodné, zde je ovšem překážkou fakt, že jsou všechny tyto technologie založené na jazyce JavaScript, se kterým sice autor nějaké zkušenosti má, nicméně pouze na úrovni programování jednoduchých funkcí, které rozšiřují uživatelské rozhraní webových aplikací vytvořených jinou technologií. Z toho důvodu autor usoudil, že by nebylo příliš efektivní se pokoušet tyto technologie používat pro vývoj tak rozsáhlého projektu, jakým je systém SOS.

Django je framework pro tvorbu webových aplikací využívající programovací jazyk Python. Implementuje architektonický vzor Model-View-Template, což je modifikace často využívaného vzoru MVC. Jádro frameworku obsahuje objektově-relační mapper mezi relační databází a datovým modelem, který je definován jako třídy v jazyka Python. Framework nabízí i další komponenty, jako například serializační a validační systém pro formuláře, cachovací systém nebo šablonovací systém. Webovou aplikaci využívající Django lze v budoucnu také relativně jednoduše rozšířit o REST API využitím open-source balíčku Django REST framework[9]. Technologii Django autor detailněji popisuje v následující kapitole.

2.14.2 PostgreSQL

Jak již bylo popsáno výše, pro systém SOS byla zvolena relační databáze. Konkrétně autor práce vybral PostgreSQL[10]. Jedná se o open-sorce technologii poskytující kompletní funkcionalitu databázového stroje. Tuto technologii autor vybral především proto, že velice dobře spolupracuje se zvoleným frameworkem Django. [32]

Kromě PostgreSQL framework nativně podporuje i další relační databáze, konkrétně se jedná o technologie MariaDB, MySQL, Oracle a SQLite. Pokud by se volba PostgreSQL časem ukázala jako nevhodná, výměna databázové technologie za jinou je pouze otázkou provedení několika málo změn v konfiguraci systému [21], samotná implementace systému na konkrétní databázové technologii závislá není. Problém s výměnou databázové technologie by nastal pouze v případě, že by se autor při implementaci rozhodl použít některé z pokročilých funkcionalit specifických pro PostgreSQL, poskytovaných balíčkem `django.contrib.postgres`.

2.14.3 Debian

Pro provoz systému SOS bude použit operační systém Linux, především pro svou poměrně jednoduchou strukturu oproti alternativě v podobě Microsoft

[7] `https://reactjs.org/`
[8] `https://vuejs.org/`
[9] `https://www.django-rest-framework.org/`
[10] `https://www.postgresql.org/`

Windows a také to, že se jedná o open-source technologii. K rozhodnutí významně přispěl i fakt, že na Fakultě informačních technologií, kde bude systém SOS provozován, je běžnou praxí používat pro provoz informačních systému operační systémy unixového typu. Konkrétně autor práce vybral linuxovou distribuci Debian 10, s níž má dlouhodobě dobré zkušenosti a která vyniká svou vysokou stabilitou a množstvím balíčků dostupných z oficiálního repozitáře.

2.14.4 NGINX

Mezi webovými servery pro provoz na operačních systémech unixového typu jsou nejpoužívanějšími technologiemi NGINX[11] a Apache[12] [33]. V obou případech se jedná o open-source řešení. Autor práce zvolil pro systém SOS právě NGINX. Oproti Apache se jedná o poměrně minimalistickou implementaci HTTP serveru využívající event-driven architekturu. Díky tomu dokáže obsloužit stejný počet požadavků jako Apache, nicméně s výrazně nižšími nároky na paměť [34]. Webový server NGINX narozdíl od Apache nedokáže sám o sobě zpracovávat dynamický obsah, v takovém případě se spoléhá na externí aplikační server.

2.14.5 Gunicorn

Gunicorn[13] je open-source implmentace WSGI, která je běžně využívána právě pro propojení webového serveru NGINX jednak s aplikačním frameworkem Django, který byl vybrán pro implementaci systému SOS, ale i s dalšími, jako jsou například Flask nebo Bottle. Implementace Gunicorn využívá tzv. worker procesů k paralelnímu obsluhování požadavků. Gunicorn byl zvolen především pro jednoduchost jeho použití s frameworkem Django, které je navíc popsáno přímo v jeho dokumentaci [35].

2.14.6 Django templates a Bootstrap

Pro frontedovou část systému autor využije šablonovací systém, který framework nabízí [14], v kombinaci s volně dostupnou CSS šablonou AdminLTE[15], využívající Bootstrap[16]. Bootstrap je knihovna napsaná v jazyce CSS, díky které lze jednoduše realizovat responzivní design webové aplikace pomocí tzv. grid systému. Knihovna také obsahuje rozličné předdefinované komponenty uživatelského rozhraní, jako jsou například tabulky, formuláře, navigace a další.

[11] `https://www.nginx.com/`
[12] `https://httpd.apache.org/`
[13] `https://gunicorn.org/`
[14] `https://docs.djangoproject.com/en/3.2/ref/templates/language/`
[15] `https://adminlte.io/`
[16] `https://getbootstrap.com/`

2.14.7 GitLab

Gitlab[17] je systém pro správu repozitářů, který navíc poskytuje řadu souvisejících funkcionalit, jako například nástroje pro správu dokumentaci (tzv. wiki). GitLab také nabízí nástroje pro realizaci CI/CD. Je dostupný buďto jako SaaS, nebo v self-hosted variantě.

GitLab je pro projekty vznikající v rámci Fakulty informačních technologií běžně využívaným řešením pro správu verzí. Fakulta využívá self-hosted variantu dostupnou na adrese `https://gitlab.fit.cvut.cz`. Vzhledem k této skutečnosti, ale také pro množství nabízených funkcionalit, které jsou podrobně popsané v oficiální dokumentaci [36], je využití systému GitLab přímočarou volbou pro vývoj systému SOS.

[17] `https://about.gitlab.com/`

Kapitola 3

Realizace

Po dokončení analýzy a návrhu bylo přistoupeno k samotné implementaci systému SOS. Tato kapitola rozebírá implementaci backendové a frontendové části systému s využitím zvoleného frameworku Django. Součástí kapitoly je i popis realizace importování dat ze systému KOS a ověřování totožnosti uživatelů prostřednictvím fakultního OAuth 2.0 autorizačního serveru. Dále se kapitola zabývá realizovaným uživatelským rozhraním a jeho rozšířeními oproti návrhu. Na závěr je uveden postup nasazení systému na fakultní server s využitím CI/CD.

3.1 Vývojové prostředí

K vývoji celého projektu autor jako operační systém autor využíval linuxovou distribuci Debian ve verzi 10.9. Jednak proto, že je to operační systém, který autor používá ke své ostatní každodenní práci na počítači a dobře jej tedy zná, ale také z toho důvodu, že se jedná o stejný operační systém, pod kterým bude na fakultním serveru systém SOS provozován. Zajištění co možná nejpodobnějšího vývojového a produkčního prostředí vytváří dobré předpoklady pro bezproblémové nasazení aplikace do provozu, jelikož se tím výrazně omezuje riziko skrytých problémů s kompatibilitou kódu či použitých knihoven s produkčním prostředím.

Pro projekty vyvíjené v programovacím jazyce Python je zvyklostí využívat tzv. virtuální prostředí [37]. To umožňuje instalaci všech balíčků a knihoven, které projekt využívá, lokálně v rámci virtuálního prostředí, nikoliv globálně na úrovni celého systému. Tím je zabráněno například kolizím mezi verzemi jednotlivých balíčků napříč více projekty vyvíjenými současně na jednom systému. K tomuto účelu autor použil open-source aplikaci Pyenv[18] spolu se zásuným modulem Pyenv-virtualenv[19], což mu umožnilo pro virtuální

[18] `https://github.com/pyenv/pyenv`
[19] `https://github.com/pyenv/pyenv-virtualenv`

prostředí definovat i konkrétní verzi samotného jazyka Python. Verze jazyka Python nainstalovaná na začátku vývoje byla 3.9.0. Python byl v průběhu vývoje projektu postupně aktualizován až do verze 3.9.4.

Tvorba zdrojových kódů aplikace probíhala ve vývojovém prostředí PyCharm[20] od společnosti JetBrains. Prostředí PyCharm nabízí řadu pokročilých funkcí, jako je například Python debugger, statická analýza kódu a integrace systému pro správu verzí. Prostředí je určeno primárně pro jazyk Python, nicméně umožňuje i editaci dalších typů souborů, jako jsou například HTML, XML a JSON dokumenty, zdrojové kódy jazyka JavaScript či shell skripty. Lze do něj také nainstalovat řadu zásuvných modulů, které přidávají pokročilé funkce specifické pro některé knihovny a frameworky, jako je například Django.

Pro formátování kódu byl využíván nástroj Black [21]. Díky tomu, že byl veškerý kód před nahráním do repozitáře automaticky formátován, bylo následně jednodušší a rychlejší výstupech verzovacího systému dohledávat a chápat změny v kódu, jelikož nenastávaly situace, kdy je mezi jednotlivými verzemi rozdíl pouze ve formátování. Zároveň se autor nemusel během programování zabývat správným formátováním kódu, což vedlo k pohodlnějšímu a rychlejšímu vývoji.

Pro vývoj projektu autor využíval lokální instalaci databázového stroje PostgreSQL ve verzi 13 a konzolový nástroj `psql`, který umožňuje připojení do terminálu databázového stroje a přímé spouštění SQL příkazů.

K účelům testování uživatelského rozhraní aplikace autor používal především webové prohlížeče Google Chrome a Mozilla Firefox. K otestování správného fungování aplikace na mobilních zařízeních sloužil jednak simulátor mobilního zařízení zabudováný do prohlížeče Google Chrome, ale také nástroj Ngrok[22], který mimo jiné umožňuje vytváření HTTP a HTTPS tunelů, například právě pro účely vzdáleného přístupu k webovým aplikacím spuštěným lokálně na počítači vývojáře. Není tedy nutné aplikaci nasazovat na produkční či testovací server. S využitím tohoto nástroje mohl autor vyvíjenou aplikaci pohodlně testovat i na svém mobilním telefonu s dotykovou obrazovkou.

3.2 Datový model

Jak již bylo uvedeno v minulé kapitole, k realizaci perzistence dat systém využívá databázový stroj PostrgreSQL. Struktura datového modelu byla definována kódem v jazyce Python, který byl frameworkem Django následně využit k vytvoření příslušných databázových tabulek.

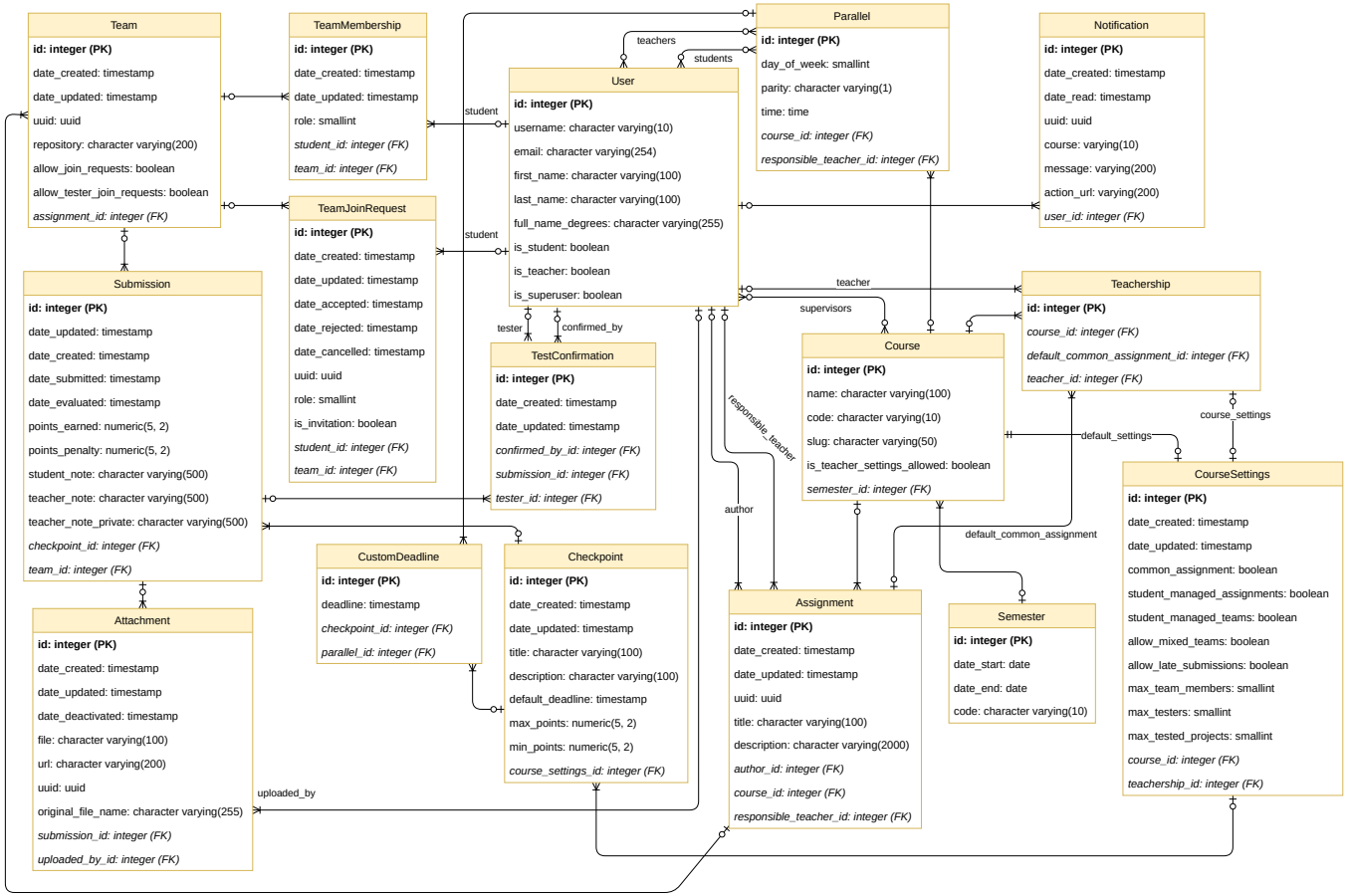
[20] `https://www.jetbrains.com/pycharm/`
[21] `https://github.com/psf/black`
[22] `https://ngrok.com/`

3.2.1 Struktura

Struktura datového modelu systému SOS je znázorněna na obrázku 3.1. Všechny M:N vazby ve schématu jsou realizovány vazebními tabulkami. Pro přehlednost byly z diagramu vynechány vazební tabulky, které neobsahují žádné vlastní atributy, ale pouze cizí klíče. Takové tabulky jsou znázorněny pouze symbolem pro M:N relaci. Konkrétně se jedná o relace `students` a `teachers` mezi entitami `User` a `Parallel` a relaci `supervisors` mezi entitami `User` a `Course`.

Systém umožňuje existenci více předmětů (model `Course`) a jejich konfiguraci včetně kontrolních bodů (`CourseSettings`, `Checkpoint`) buďto garantem předmětu (relace `default_settings`), nebo přímo jednotlivými vyučujícími (relace `course_settings`). Záznamy pro jednotlivé předměty jsou pro každý semestr vytvářeny znovu, v databázi se tedy může vyskytovat více předmětů se stejným názvem a kódem, ale každý bude mít vazbu na jiný semestr (model `Semester`). Jednotlivé paralelky jsou reprezentovány modelem `Parallel`.

Zadání (model `Assignment`) mohou vytvářet jak studenti, tak učitelé. To platí i o týmech (model `Team`), které vzniknou k daným zadáním. Do týmu mohou být studenti buďto přizváni vedoucím či vyučujícím, nebo mohou sami požádat o přijetí do týmu. V obou případech toto realizuje model `TeamJoinRequest`. Odevzdání vypracovaného řešení je uchovávají modely `Submission` a `Attachment`. `Submission` v sobě udržuje jak samotné vypracované řešení, tak i hodnocení vyučujícího. `Attachment` může obsahovat buďto webový odkaz na vypracované řešení (atribut `url`), nebo cestu k souboru nahranému studentem, který je uložen na disku (atribut `file`). Aplikační logika zajišťuje, že je vždy vyplněn právě jeden z těchto atributů. Potvrzování otestování řešení studenty reprezentuje model `TestConfirmation`, který uchovává odkazy na studenta, který testoval, na studenta, který testování potvrdil a na odevzdané řešení, jež bylo předmětem testování.



Obrázek 3.1: ER diagram databáze systému SOS

3.2.2 Django ORM

Framework Django ve svém jádru obsahuje ORM – objektově-relační mapper. Ten slouží k abstrakci SQL příkazů, které jsou generovány a spouštěny na základě kódu v jazyce Python.

3.2.2.1 Model

Modely [38] jsou reprezentace struktury a chování jednotlivých datových entit. Jedná se o třídy jazyka Python, které jsou potomky třídy `django.db.models.Model`. Každá třída, která je modelem, je mapována na určitou tabulku v databázi. Atributy modelu reprezentují jednotlivé sloupce v tabulce. Model dále může obsahovat vnitřní třídu `Meta`, ve které lze definovat různé metainformace, jako jsou například definice indexů, nebo způsob řazení jednotlivých záznamů v databázi.

Speciálním atributem je atribut `objects`. Ten nedefinuje sloupec tabulky, ale tzv. `Manager`, což je rozhraní sloužící k dotazování do databáze. Pokud atribut `objects` není definován, je použit výchozí `Manager`. Často je ale vhodné použít vlastní `Manager` a v něm definovat často používané dotazy, aby je nebylo nutné v kódu opakovat. Alternativně lze také definovat `QuerySet`, což je třída reprezentující kolekci záznamů v databázi, ze které lze `Manager` vytvořit zavoláním třídní metody `QuerySet.as_manager()` [39].

V ukázkách 1 a 2 je jako příklad uvedena implementace reprezentace zadání projektu, tedy modelu `Assignment` spolu s rozhraním `AssignmentQuerySet`.

3.2.2.2 Databázové migrace

Pomocí tzv. migrací [40] Django umožňuje propagaci změn provedených v jednotlivých modelech do databázového schématu. Není tedy potřeba po provedení změn databázi mazat a vytvářet znovu, ani do ní manuálně prostřednictvím SQL příkazů. Migrace po jejich vytvoření provádět oběma směry. Pokud se tedy při vývoji zjistí, že provedené změny nebyly vhodné a je potřeba databázové schéma uvést do původního stavu, lze tak učinit spuštěním jednoho příkazu.

Migrace jsou frameworkem generovány ve formě souborů obsahujících kód v jazyce Python, jenž určuje operace, které mají být provedeny. Migrační soubory se pak uchovávají ve verzovacím systému spolu s ostatními zdrojovými kódy, což umožňuje jednoduché vytváření a aktualizace databázového schématu jak na počítačích vývojářu, tak i na testovacích a produkčních serverech.

Existují dva druhy migrací – migrace schématu a migrace dat. Migrace schématu mění samotné databázové schéma, tedy například přidávají, odstraňují či upravují jednotlivé sloupce nebo rovnou celé tabulky. V naprosté většině případů pro vytvoření tohoto typu migrací vývojář pouze spustí příkaz pro jejich vygenerování a do takto vzniklých souboru již dále nezasahuje.

```python
class Assignment(TimestampedModelMixin, models.Model):
    objects = AssignmentQuerySet.as_manager()

    # fields
    uuid = models.UUIDField(
        default=uuid.uuid4,
        unique=True,
        editable=False,
    )

    title = models.CharField(max_length=100)
    description = models.CharField(max_length=2000)

    # relations
    author = models.ForeignKey(
        settings.AUTH_USER_MODEL,
        on_delete=models.SET_NULL,
        related_name="created_assignments",
        null=True,
        blank=True,
    )
    responsible_teacher = models.ForeignKey(
        settings.AUTH_USER_MODEL,
        on_delete=models.RESTRICT,
        related_name="responsible_assignments",
    )
    course = models.ForeignKey(
        "courses.Course",
        on_delete=models.CASCADE,
        related_name="assignments",
    )

    def __str__(self):
        return self.title

    def get_absolute_url(self):
        return reverse_for_course(
            self.course,
            "assignments:detail",
            kwargs={"assignment": self.uuid},
        )
```

Listing 1: Ukázka kódu modelu `Assignment`

```
class AssignmentQuerySet(models.QuerySet):
    def available_for_user(self, user, course):
        avail = self.model.objects.filter(course=course)
        if course.is_user_teacher(user):
            return avail.filter(author=user)
        if course.is_user_student(user):
            parallel = course.get_parallel_for_student(user)
            course_settings = course.get_settings_for_teacher(
                parallel.responsible_teacher
            )
            if course_settings.allow_mixed_teams:
                return avail.filter(author__taught_courses=course)
            return avail.filter(author__in=parallel.teachers.all())
        return self.model.objects.none()
```

Listing 2: Ukázka kódu třídy `AssignmentQuerySet`

Datové migrace nemění schéma, ale upravují data, která jsou v databázi uložena. Využívají se převážně v případech, kdy dochází k takovým změnám schématu, že není jednoznačné, jakým způsobem by měla být uložena stávající data – například pokud mají být záznamy uložené v jedné tabulce rozděleny do dvou nově vzniklých tabulek. Při vytváření datových migrací vývojář spuštěním příkazu vygeneruje tzv. prázdnou migraci a následně v takto vzniklém souboru definuje operace, které mají být s daty provedeny.

3.3 Struktura aplikace

V této části autor popisuje, jakým způsobem je aplikace strukturována do jednotlivých komponent a jak spolu tyto komponenty spolupracují. Nejprve rozebírá, jakým způsobem jsou obvykle strukturovány projekty využívající framework Django a následně se zabývá implementacemi jednotlivých komponent v rámci systému SOS.

3.3.1 Implementace architektury MVT

Jak již autor zmínil v analytické části práce, framework Django implementuje návrhový vzor MVT, neboli Model-View-Template. Principy fungování jednotlivých kompoment této architektury jsou popsány v následujících podsekcích.

3.3.1.1 Směrování požadavků

Vstupním bodem do aplikace je směrovač požadavků realizovaný souborem `urls.py`. Prostřednictvím tohoto směrovače server rozhoduje, kterému konkrétnímu View má být příchozí HTTP požadavek předán ke zpracování. V souboru jsou uvedené jednotlivé URI adresy spolu s odpovídajícím View. Pro přehlednost je zvykem tuto konfiguraci strukturovat podle jednotlivých částí systému, aplikace má v takovém případě jeden kořenový soubor `urls.py` a několik dalších takových souborů pro jednotlivé části aplikace, na které je v tomto kořenovém souboru odkazováno zavoláním funkce `django.urls.include()` [41]. Struktura směrování může mít více úrovní. Toto je ilustrováno ukázkami 3 a 4.

```
urlpatterns = [
    path("", HomepageView.as_view(), name="home"),
    path("admin/doc/",
         include("django.contrib.admindocs.urls")),
    path("admin/", admin.site.urls),
    path("login/", LoginView.as_view(), name="login"),
    path("logout/", LogoutView.as_view(), name="logout"),
    path("login-as/", include("apps.users.loginas.urls",
                              namespace="login_as")),
    path("oauth/", include("apps.oauth.urls",
                           namespace="oauth")),
    path("", include("apps.homepage.urls",
                     namespace="homepage")),
    path("courses/", include("apps.courses.urls",
                             namespace="courses")),
    path("submissions/", include("apps.submissions.urls",
                                 namespace="submissions")),
    path("attachments/", include("apps.attachments.urls",
                                 namespace="attachments")),
    path("join/", include("apps.teams.urls.join",
                          namespace="team_join")),
    path("notifications/", include("apps.notifications.urls",
                                   namespace="notifications")),
]
```

Listing 3: Konfigurace směrování v kořenovém souboru `urls.py`

```
urlpatterns = [
    path("", views.AssignmentListView.as_view(), name="list"),
    path("create/", views.AssignmentCreateView.as_view(),
         name="create"),
    path("<uuid:assignment>/", include([
         path("", views.AssignmentDetailView.as_view(),
              name="detail"),
         path("update/", views.AssignmentUpdateView.as_view(),
              name="update")
    ])),
]
```

Listing 4: Konfigurace směrování v souboru `assignments/urls.py`

3.3.1.2 View

Směrovač určí, který View má daný požadavek zpracovávat a předá mu jej. View je zodpovědný za provedení požadovaných akcí a výsledkem jeho činnosti je odpověď, která bude odeslána klientovi. View jsou realizovány buďto funkcemi, nebo třídami, které jsou na funkce předevedny voláním třídní metody `View.as_view()` v souboru `urls.py` [42].

Při vývoji systému SOS se autor rozhodl pro implementaci všech View pomocí tříd. Tento přístup umožňuje lepší strukturování logiky v rámci jednotlivých View a také využití řady předdefinovaných generických View určených pro běžné akce, jako je například zobrazení seznamu objektů určitého typu nebo jejich vytváření a úpravy uživatelem. [43]

S využitím objektového přístupu lze také definovat některé funkcionality společné pro více View pomocí tříd typu *mixin*, z nichž může defininovaný View dědit. Jako příklad autor v ukázce 5 uvádí implementaci třídy `AssignmentCreateView`, která slouží k vytvoření nového zadání projektu uživatelem prostřednicvím formuláře. Tato třída je potomkem generického `CreateView`, který je součástí frameworku a mimo jiné využívá například frameworkem poskytovaný `LoginRequiredMixin`, který zajišťuje, že uživateli, který není do systému přihlášený, není povolen přístup k funkcionalitě a je místo toho přesměrován na přihlášení.

Jak je patrné z ukázky kódu, využitím generického `CreateView` je vývojáři ušetřena téměř veškerá práce s vytvářením nového objektu. Pouze definuje model, který má být vytvořen (atribut `model`), formulář, který má být zobrazen uživateli (atribut `form_class`, viz sekce 3.3.1.5) a šablonu, která má být použita k vygenerování HTML dokumentu odeslaného klientovi (atribut `template_name`, viz sekce 3.3.1.4). Veškerá logika související s tímto procesem (zobrazení formuláře uživateli, validace vyplněného formuláře, vytvoření

záznamu v databázi a přesměrování uživatele na další stránku) je definována v generickém `CreateView`, protože je z valné většiny společná pro vytváření všech typů objektů [44].

```
class AssignmentCreateView(
    LoginRequiredMixin,
    UserPassesTestMixin,
    CourseMixin,
    BreadcrumbMixin,
    CreateView,
):
    model = Assignment
    form_class = AssignmentCreateForm
    template_name = "assignments/create.html"

    def test_func(self):
        return self.is_user_teacher

    def get_breadcrumb_items(self):
        return super().get_breadcrumb_items() + [
            self.course.breadcrumb_item,
            ("Assignments",
             self.course.get_assignment_list_url()),
            ("New", ""),
        ]

    def get_form_kwargs(self):
        kwargs = super().get_form_kwargs()
        kwargs["author"] = self.request.user
        kwargs["responsible_teacher"] = self.request.user
        kwargs["course"] = self.course
        return kwargs
```

Listing 5: Implementace třídy `AssignmentCreateView`

3.3.1.3 Model

Pro přístup k datům slouží třídy typu Model, což jsou třídy, které jsou potomky třídy `django.db.models.Model`. Principy fungování těchto tříd již byly popsány v sekci 3.2.2.1. Mimo samotnou strukturu objektů lze v modelech definovat i aplikační logiku, která souvisí s daným modelem. Operace, které se vztahují jeden konkrétní objekt (tedy na jeden řádek tabulky), se defi-

nují prostřednictvím metod v těchto třídách. Operace prováděné nad celou tabulkou jsou realizovány pomocí tzv. manažerských metod ve třídě typu Manager (potomek třídy `django.db.models.Manager`) nebo QuerySet (potomek `django.db.models.QuerySet`), která je Modelu přidělena atributem `objects`.

3.3.1.4 Template

K vygenerování odpovědi, která má být odeslána klientovi, využívá Django tzv. Templates, což textové soubory využívající syntaxi Django template language. Tyto soubory jsou využívány převážně ke generování HTML dokumentů, ale lze je použít k vygenerování libovolného typu textového souboru, například XML, CSV nebo i kódu v jazyce JavaScript.

Template obsahuje proměnné, které jsou při generování nahrazeny hodnotami a tzv. *tags*, které plní roli základních programovacích konstruktů, jako je testování pravdivostní hodnoty (tag `if`) nebo iterování přes kolekce (tag `for`). Django template language obsahuje řadu předdefinovaných konstruktů, vývojář navíc může vytvářet i vlastní. Nicméně není možné přímo spouštět libovolný kód v rámci Template – jsou určené pouze k prezentaci dat, nikoliv k realizaci aplikační logiky.

Jednou z nejužitečnějších vlastností Django template language je dědičnost. Umožňuje například vytvořit základní šablonu, která definuje kostru dokumentu a tzv. bloky (tag `block`). Další šablony potom mohou základní šablonu načíst tagem (`extends`) a pouze vyplnit tyto bloky svým specifickým obsahem. Šablony lze také vkládat do sebe pomocí tagu `include`. [45]

V ukázce 6 autor uvádí šablonu `submission.html`, sloužící ke generování HTML reprezentace informací o odevzdání vypracovaného řešení spolu s přílohami.

3.3.1.5 Form

Formuláře (anglicky *form*) jsou třídy, které jsou potomky třídy `django.forms.Form`. Formuláře primárně slouží k validaci dat, která vstupují do systému. Ve formuláři lze definovat jednotlivá pole společně s jejich datovými typy a požadavky, které musí vložená data splňovat. Další využití takové třídy je kromě samotné validace vstupních dat i samotné zobrazení formuláře uživateli. Děje se tak vložením instance formuláře jako proměnné do šablony, kde je následně převedena na HTML reprezentace a vložena do dokumentu. [46]

Speciálním druhem formuláře je třída `django.forms.ModelForm`, která kromě výše zmíněných funkcionalit pracuje i se samotnou instancí nějakého modelu. Slouží buďto k zadání hodnot při vytváření instance Modelu, nebo k jejich změnám. Takový formulář má navíc metodu `ModelForm.save()`, po jejímž zavolání je provedena validace vstupních dat i vzhledem k definici Modelu a pokud byla úspěšná, je instance Modelu uložena do databáze.

```
<p>
    <b>Date submitted:</b>
    {{ submission.date_submitted|default_if_none:"-" }}<br>

    <b>Status:</b>
    {{ submission.status.label }}
</p>
<p>
    <b>Student note</b><br>
    {{ submission.student_note|linebreaksbr }}
</p>

<hr>

<h5>Attachments</h5>
{% for attachment in submission.attachments.active %}
    <a href="{{ attachment.attachment_url }}" target="_blank"
       class="btn btn-block btn-outline-secondary text-left">
        {{ attachment }}
    </a>
{% endfor %}
```

Listing 6: Šablona `submission.html`

Jako příklad autor v ukázce 7 uvádí formulář `AssignmentCreateForm`, který je potomkem třídy `django.forms.ModelForm` a slouží k vytváření nového zadání vyučujícím či studentem.

3.3.2 Struktura projektu

Při vytvoření nového projektu framework vytvoří výchozí adresářovou strukutru, kterou může vývojář následně mírně upravovat. Názory vývojářů na best-practices ohledně struktury Django projektu se různí, nicméně základní principy zůstávají vždy stejné. [47, kap. 3] Autor zde popisuje adresářovou strukturu, kterou zvolil pro systém SOS na základě svých zkušeností.

V kořenovém adresáři projektu se nachází soubor `manage.py`, což je Python skript, který slouží ke spouštění serverových příkazů, jako například `python manage.py runserver`, který spustí server, nebo `python manage.py makemigrations` k vytváření databázových migrací.

V adresáři `conf/` se nachází modul `settings`, který obsahuje výchozí konfiguraci projektu a může obsahovat i další specifické konfigurace, jako například pro spuštění na produkčním serveru, či lokální spuštění na systému vý-

```
sos/
├── apps/
│   ├── assignments/
│   ├── attachments/
│   ├── checkpoints/
│   ├── courses/
│   ├── homepage/
│   ├── notifications/
│   ├── oauth/
│   ├── submissions/
│   ├── teams/
│   ├── users/
│   └── __init__.py
├── conf/
│   ├── settings/
│   │   ├── ci.py
│   │   ├── base.py
│   │   ├── local.py
│   │   ├── local.tmpl.py
│   │   ├── production.py
│   │   └── stage.py
│   ├── __init__.py
│   ├── urls.py
│   └── wsgi.py
├── core/
│   ├── context_processors.py
│   ├── fields.py
│   ├── forms.py
│   ├── mixins.py
│   ├── storage.py
│   ├── utils.py
│   └── views.py
├── locale/
├── requirements/
│   ├── base.txt
│   ├── local.tmpl.txt
│   └── local.txt
├── staticfiles/
├── templates/
├── .env
├── .env.tmpl
└── manage.py
```

Obrázek 3.2: Adresářová struktura projektu SOS

```python
class AssignmentCreateForm(forms.ModelForm):
    class Meta:
        model = Assignment
        fields = ("title", "description")
        widgets = {"description": forms.Textarea}

    def __init__(self, author, responsible_teacher, course,
                 *args, **kwargs):
        super().__init__(*args, **kwargs)
        self.author = author
        self.responsible_teacher = responsible_teacher
        self.course = course

    def save(self, commit=True):
        self.instance.author = self.author
        self.instance.responsible_teacher = (
            self.responsible_teacher
        )
        self.instance.course = self.course
        return super().save(commit=commit)
```

Listing 7: Implementace formuláře `AssignmentCreateForm`

vojáře [47, kap. 5.2]. Lokální konfigurace není uložena ve verzovacím systému, jelikož může být pro každého vývojáře odlišná. Ve verzovacím systému je uložena pouze šablona `local.tmpl.py`, ze které si každý vývojář vytvoří svou lokální konfiguraci a uloží ji pod názvem `local.py`. Dále adresář `conf/` obsahuje soubor `urls.py`, ve kterém je definována kořenová konfigurace směrovače příchozích požadavků a soubor `wsgi.py`, který byl vygenerován frameworkem a slouží k integraci aplikace s rozhraním WSGI při spuštění v produkčním prostředí.

V adresáři `requirements/` jsou uloženy textové soubory, ve kterých jsou definovány balíčky, které je potřeba nainstalovat do virtuálního prostředí pro spuštění projektu. Opět je zde více možností, například pro instalaci na produkčním serveru a pro vývojové prostředí.

V adresáři `apps/` se nachází jednotlivé aplikace. U projektů používajících framework Django je zvykem členit projekt do více tzv. aplikací ve formě Python modulů. Každá aplikace je zodpovědná za určitou logickou část funkcionalit systému. Modul aplikace v sobě obsahuje příslušné modely, formuláře, aplikační logiku, dílčí konfiguraci směrovače požadavků a šablony pro generování odpovědí. Adresářová struktura aplikace je ukázána na ukázce 3.3 na příkladu aplikace `checkpoints`, která implementuje logiku související s kon-

```
checkpoints/
├── migrations/ ..... databázové migrace
├── templates/ ..... šablony pro generování HTML dokumentů
├── tests/ ..... unit testy
├── __init__.py
├── admin.py ..... konfigurace administrátorského rozhraní
├── apps.py ..... konfigurace aplikace
├── forms.py ..... implementace formulářů
├── models.py ..... implementace modelů
├── signals.py
├── urls.py ..... konfigurace směrovače požadavků
└── views.py ..... implementace Views
```

Obrázek 3.3: Adresářová struktura aplikace `checkpoints`

trolními body odevzdávání vypracovaných řešení v systému SOS. Aplikace obecně nemusí obsahovat všechny uvedené položky, nebo mohou naopak obsahovat i řadu dalších, v závislosti na požadovaných funkcionalitách a rozhodnutích vývojáře [47, kap 4.4].

Adresář `core/` má podobný obsah a strukturu jako jednotlivé aplikace. Nachází se zde například implementace pomocných funkcí a další položky, které přímo nesouvisí s žádnou logickou částí systému a nelze je tedy smysluplně zařadit do jedné z aplikací.

V kořenovém adresáři se nakonec nacházejí ještě adresáře `locale/`, `staticfiles/` a `templates/`. V adresáři `locale/` se nachází textové soubory s jazykovými lokalizacemi všech řetězců, které jsou zobrazovány v uživatelském rozhraní. V adresáři `staticfiles` jsou uloženy statické soubory, na které jsou vkládány do jednotlivých stránek aplikace. Jedná se například o obrázky, soubory s CSS styly a JavaScript. Adresář `templates/` obsahuje šablony pro generování HTML dokumentů, stejně jako stejnojmenné adresáře v jednotlivých aplikacích. Podobně jako v adresáři `core/` jsou zde uloženy šablony, které logicky nesouvisí s žádnou konkrétní aplikací, ale patří k projektu jako takovému, tedy například definice některých komponent uživatelského rozhraní využívaných napříč celým systémem.

3.3.2.1 Aplikace Assignments

Aplikace `assignments` v systému realizuje zadání projektů. Obsahuje jediný model `Assignment` jakožto reprezentaci zadání a příslušné Views a formuláře pro vytváření, zobrazování a úpravy jednotlivých zadání.

3.3.2.2 Aplikace Attachments

Aplikace `attachments` realizuje přílohy, které studenti nahrávají ke svým odevzdávaným řešením. V aplikaci je jediný model `Attachment`, který je reprezentací přílohy. V modelu je uložen odkaz na nahraný soubor, nebo URL adresa, v případě, že se příloha nachází někde na internetu, mimo systém SOS. Dále také informace o tom kdo a kdy soubor nahrál a ke kterému odevzdání příloha patří. Součástí aplikace je také `AttachmentProtectedServeView`, který umožňuje zobrazení či stažení přílohy a zajišťuje, že tak budou schopni učinit pouze oprávnění uživatelé.

3.3.2.3 Aplikace Checkpoints

Aplikace `checkpoints` implementuje kontrolní body odevzdávání projektů. Její součástí jsou dva modely – `Checkpoint`, jenž reprezentuje samotný kontrolní bod se všemi souvisejícími informacemi a `CustomDeadline`, který udržuje informaci o nastaveném termínu odevzdání pro konkrétní paraleku, jelikož tyto termíny se pro jednotlivé paralelky mohou navzájem lišit. Aplikace obsahuje také příslušné Views a formuláře pro vytváření a úpravy kontrolních bodů a termínů odevzdání.

3.3.2.4 Aplikace Courses

Aplikace `courses` je asi nejrozsáhlejší aplikací v rámci systému SOS. Obsahuje modely reprezentující jednotlivé semestry, předměty a paralelky a také konfigurace předmětů pro jednotlivé vyučující, spolu se souvisejícími Views a formuláři. Součástí aplikace je také třída `CourseMixin`, což je mixin, který používá řada Views napříč celým systémem. Zajišťuje přístup k informacím o předmětu, ke kterému se vztahuje objekt (např. tým, zadání či kontrolní bod) se kterým se pracuje a poskytuje pomocné metody vztahující se k předmětu, například metodu `CourseMixin.is_user_student()` určenou k rozpoznání, zda je uživatel ve vybraném semestru studentem daného předmětu.

3.3.2.5 Aplikace Homepage

Aplikace `homepage` je naopak asi nejmenší v celém systému. Neobsahuje žádné modely, ale pouze trojici Views pro zobrazení domovské stránky – jeden generický a po jednom pro studenta a pro vyučujícího.

3.3.2.6 Aplikace Notifications

Aplikace `notifications` slouží k realizaci upozornění uživatelů na důležité události v systému SOS, které se týkají tvorby týmů a odevzdávání vypracovaných řešení a jejich ohodnocování. Aplikace obsahuje jeden model `Notification`, který udržuje obsah upozornění, URL adresu stránky, na kterou upozornění odkazuje (např. stránka s ohodnoceným odevzdáním) a informaci o

tom, zda si již uživatel notifikaci zobrazil. Aplikace dále obsahuje dva Views, jeden pro zobrazení seznamu všech upozornění a druhý, který zaznamená informaci o tom, že byla notifikace zobrazena a přesměruje uživatele na relevantní stránku.

3.3.2.7 Aplikace OAuth

Aplikace `oauth` se liší od ostatních aplikací tím, že v ní nejsou žádné modely ani Views či formuláře sloužící pro práci s objekty v systému SOS. Tato aplikace přináší do systému funckionalitu autentizace uživatelů prostřednictvím fakultního OAuth 2.0 serveru a přístup k datům ze systému KOS prostřednictvím API. Také obsahuje Views sloužící k přihlášení uživatele prostřednictvím fakultního OAuth 2.0 serveru, případně registraci uživatele, který ještě v systému SOS uložen není. Fungování této aplikace autor podrobněji rozebírá v sekci 3.4.

3.3.2.8 Aplikace Submissions

Účelem aplikace `submissions` je realizace odevzdávání vypracovaných řešení a jejich testování a hodnocení. Obsahuje dva modely – `Submission`, který reprezentuje samotné odevzdání a `TestConfirmation`, což je záznam o otestování řešení studentem. Součástí aplikace jsou Views a formuláře určené k odevzdávání řešení, spravování a úpravám odevzdání studenty, potvrzování otestování řešení studenty a k hodnocení řešení vyučujícími.

3.3.2.9 Aplikace Teams

Aplikace `teams` je určená ke správě týmů. Obsahuje mimo jiné modely `Team` reprezentující samotný tým a `TeamJoinRequest`, který představuje buďto pozvánku do týmu, nebo žádost o přijetí do týmu. Součástí aplikace je řada Views a formulářů pro různé akce související s týmy, jako jsou vytvoření týmu a správa členů a nastavení týmu.

3.3.2.10 Aplikace Users

Aplikace `users` obsahuje jediný model `User`, který je potomkem třídy `django.contrib.auth.models.AbstractUser`. Tento model je reprezentací uživatele v systému. Poskytuje veškeré funkcionality, jako standartní reprezentace uživatele poskytovaná frameworkem Django [48] a přidává navíc řadu dalších, které jsou pro systém SOS potřeba. Jedná se například o možnost vytváření uživatelů z dat získaných z fakultních informačních systémů, rozlišení uživatelů na studenty a vyučující a přístup k informacím o studovaných či vyučovaných předmětech a týmech, jejichž je daný uživatel členem.

```
oauth/
├── utils/
│   ├── kos/
│   │   ├── data.py ..... funkce pro získávání dat ze systému KOS
│   │   └── xml.py ..... třída pro zpracování XML dokumentu
│   └── token.py ..... funkce pro práci s access tokenem
├── conf.py ..... konfigurace aplikace
├── exceptions.py ..... definice souvisejících výjimek
├── tests.py ..... unit testy
├── urls.py ..... konfigurace směrovače požadavků
└── views.py ..... implementace Views
```

Obrázek 3.4: Adresářová struktura aplikace `oauth`

3.4 Integrace s fakultními systémy

V této sekci autor popisuje, jak je v systému SOS realizováno ověřování identity a přihlašování uživatelů prostřednictvím fakultního OAuth 2.0 autorizačního serveru a také získávání dat ze systému KOS prostřednictvím API. V rámci popisu se autor odkazuje na jednotlivé soubory a moduly aplikace `oauth`, jejíž strukturu znázorňuje ukázka 3.4. Z ukázky jsou pro přehlednost vypuštěny některé soubory, které neobsahují kód relevantní pro pochopení fungování aplikace, ale jsou vyžadovány implementací samotného frameworku Django.

3.4.1 OAuth 2.0

Jak již bylo zmíněno v analytické části, fungování protokolu OAuth 2.0 je detailně zdokumentováno standardem RFC 6749 [27]. Podrobná dokumentace k použití fakultního autorizačního serveru je dostupná v repozitáři ve službě GitHub, kde se zároveň nachází i zdrojové kódy samotného serveru. Je zde popsán i celý proces ověření identity uživatele [49]. V této sekci autor popisuje, jak je konkrétně přihlášení uživatele s využitím protokolu OAuth 2.0 implementováno v systému SOS.

Proces začíná přesměrováním uživatele na příslušnou URL adresu autorizačního serveru s nastavenými příslušnými parametry. Jedním z nich je například parametr `state`, který slouží jako bezpečnostní prvek. Jedná se o náhodně vygenerovanou hodnotu velikosti 1024 bitů, která je spolu s dalšími parametry podepsána algoritmem HMAC-SHA256. Tato hodnota je také uložena do aktuální *session* pro pozdější kontrolu. Fungování *session* ve frameworku Django je popsáno v sekci 3.5.6. Jedním z dalších parametrů je `scope`, který určuje, k jakým datům bude mít aplikace po úspěšném získání autorizace přístup. Systém SOS si vystačí s rámcem `urn:ctu:oauth:kosapi:public.readonly`, který poskytuje všem uživatelům v rámci univerzity přístup k rozvrhovým da-

tům. Sestavení URL adresy s příslušnými parametry a přesměrování uživatele zajišťuje třída `OAuthRedirectView`.

Autorizační server provede ověření identity uživatele a následně jej přesměruje zpět do systému SOS, konkrétně na adresu `/oauth/callback/`. Zpracování požadavku na tuto adresu zajišťuje třída `OAuthCallbackView`. Nejprve je ověřeno, zda byla autorizace úspěšná a požadavek neobsahuje chybu. Také je ověřena legitimita požadavku na základě parametru `state`. V případě, že je vše dosud proběhlo správně, postupuje se dále tak, jak je uvedeno v dokumentaci autorizačního serveru. Výsledkem procesu je uživatelské jméno a validní *access token* vystavený pro daného uživatele.

Podle získaného uživatelského jména je v databázi nalezen příslušný uživatel. Pokud uživatel s tímto uživatelským jménem zatím neexistuje, systém využije *access token* k získání dat o uživateli ze systému KOS a vytvoří záznam o uživateli v databázi. Následně je uživatel přihlášen a přesměrován na domovskou stránku, případně na adresu definovanou URL parametrem `next` v původním požadavku, který zahájil proces přihlašování.

Po přihlášení uživatele je navíc *access token* a spolu s ním i čas jeho expirace a tzv. *refresh token* uložen do *session* přihlášeného uživatele. Access token tak může být později opět použit k získání dat ze systému KOS. Pokud *access token* expiruje, je za pomocí *refresh tokenu* získán od autorizačního serveru nový. Všechny funkcionality, které souvisí s *access tokenem* jsou realizovány funkcemi v souboru `token.py`.

3.4.2 KOS

K získávání dat ze systému KOS slouží modul `kos` v aplikaci `oauth`. Tento modul je skládá ze dvou souborů – `xml.py` a `data.py`.

KOSapi poskytuje veškerá data pouze ve formátu XML. Soubor `xml.py` obsahuje třídu `XMLNode`, která slouží ke zpracování dat v XML dokumentu pro potřeby systému SOS. Ke čtení obsahu XML dokumentu je využit balíček `xml.dom.minidom`, který je součístí standartní knihovny jazyka Python.

Soubor `data.py` obsahuje funkce, které slouží k získávání dat ze systému KOS. Nejdůležitější funkcí je `_retrieve_resource()`, která s využitím *access tokenu* získá z KOSapi požadovaná data ve formě XML dokumentu. Datové zdroje poskytované KOSapi jsou popsány v jeho dokumentaci [50]. Takto získaná data jsou pak v dalších funkcích zpracována pomocí třídy `XMLNode` a převedena do podoby Python objektů, které jsou využívány ve Views, ze kterých jsou tyto funkce volány.

Tyto funkce jsou využívány především na dvou místech. Prvním z nich je `OAuthCallbackView`, kde je v případě, že uživatel zatím není zaznamenán v databázi, použita funkce `get_user_info()` k získání potřebných dat o tomto uživateli.

Druhým místem je `CourseCreateView` v aplikaci `courses`. Tento View slouží k zobrazení seznamu předmětů, které může přihlášený uživatel (vyu-

čující) vytvořit a následnému vytvoření profilu předmětu, který zvolí. Nejprve je použita funkce `get_supervised_courses()` k získání seznamu předmětů, jejichž je uživatel garantem. Po výběru předmětu jsou použity funkce `get_supervisor_usernames()`, `get_user_info()` a `get_parallels()` k získání informací o ostantích garantech předmětu a o jednotlivých paralelkách a jejich studentech a vyučujících. Tyto informace jsou následně uloženy do databáze, čímž je profil předmětu vytvořen. Současně jsou v tuto chvíli vytvořeny i profily uživatelů, kteří jsou studenty nebo vyučujícími daného předmětu a zatím v systému SOS nejsou.

Ke snížení objemu přenášených dat a zrychlení celého procesu při dotazování na KOSapi byl použit filtr XPartial, který slouží k určení pouze takové podmnožiny datového zdroje, která je potřeba. Datové zdroje v KOSapi totiž obsahují řadu metainformací a položek, které nejsou pro účely systému SOS relevantní. Syntaxe a použití XPartial jsou popsány v dokumentaci KOSapi [51].

3.5 Zabezpečení

Framework Django nabízí řadu hotových bezpečnostních prvků, které není ve většině případů třeba nijak složitě konfigurovat. Pro jejich správné fungování je ovšem potřeba o nich vědět a používat framework takovým způsobem, aby nebylo jejich fungování narušeno. Tyto prvky popisuje především oficiální dokumentace frameworku [52], ale také je spolu s doporučeními pro správné používání přehledně shrnuje publikace Two Scoops of Django [47, kap. 28]. Mezi tyto bezpečnostní prvky se řadí především ochrana proti XSS útokům, CSRF útokům, SQL injection, používání SSL/HTTPS a další. V následujících podsekcích autor shrnuje použití těchto bezpečnostních prvků v systému SOS.

3.5.1 Ochrana proti XSS

XSS neboli *Cross site scripting* je útok, při kterém je do webové stránky zobrazované uživateli útočníkem vložen škodlivý JavaScript, který je následně v prohlížeči uživatele spuštěn. Šablonovací systém frameworku Django brání provedení těchto útoků tím, že veškerá data vkládaná do šablon automaticky ošetřuje, především nahrazením určitých znaků potřebných pro tvorbu HTML kódu za jejich bezpečné alternativy. Oficiální dokumentace [45] říká, že se konkrétně jedná o následující znaky:

- znak `<` je nahrazen za `<`
- znak `>` je nahrazen za `>`
- znak `'` je nahrazen za `&#x27;`
- znak `"` je nahrazen za `&quot;`

- znak `&` je nahrazen za `&`

Tím je zabráněno vložení HTML tagu `<script>` a tedy i spuštění škodlivého JavaScriptu.

Tuto ochranu může vývojář narušit především špatné použití šablonovacího systému, při kterém umožní vložit do stránky JavaScript bez použití tagu `<script>`, jako například v tomto příkladu:

```
<style class={{ var }}>priklad</style>
```

Díky absenci uvozovek kolem vložené proměnné `var` může být například při její hodnotě `class1 onmouseover=javascript:func()` JavaScript spuštěn. V tomto případě lze problém vyřešit přidáním uvozovek okolo proměnné.

Šablonovací systém také umožňuje vypnutí nahrazování nebezpečných znaků pro konkrétní proměnnou, což může být v určitých přípaech užitečené. V systému SOS je toto využíváno při zobrazování popisu zadání a kontrolních bodů, které jsou uživatelem definovány v textovém editoru Summernote[23]. Je tedy potřeba zabránit vložení tagu `<script>` a dalších nechtěných elementů jiným způsobem. Autor práce se rozhodl k tomuto využít balíček `django-bleach`[24], který umožňuje jednoduchým způsobem specifikovat pouze některé HTML tagy, atributy a CSS třídy, které mohou být do šablony vloženy. Prvky, které nejsou explicitně označeny jako bezpečné jsou pak z vkládaného textu odstraněny, čímž je problém vyřešen. Povolené prvky v systému SOS jsou například nadpisy, tabulky, seznamy nebo styly pro změnu barvy a velikosti písma.

3.5.2 Ochrana proti CSRF

CSRF neboli *Cross site request forgery* je útok, při kterém je provedena nezamýšlená akce kliknutím na odkaz na webu útočníka, který odešle požadavek do systému, jenž je cílem útoku. Systém v takovém případě předpokládá, že se jedná o legitimní požadavek odeslaný uživatelem, protože tento požadavek obsahuje příslušné cookies, které jsou uloženy v prohlížeči uživatele.

Proti tomuto typu útoku se systém SOS brání dvěma způsoby. Jedním je zajištění, že zpracování HTTP požadavků na tzv. bezpečné metody (především metoda GET) [53, sekce 4.2.1] nebude mít žádné vedlejší efekty a nepovede k žádným změnám v systému.

Druhým způsobem ochrany je využívání `CsrfViewMiddleware` poskytovaného frameworkem, sloužícího k zabezpečení požadavků na metodu POST (a případně další metody, které mohou vyvolat vedlejší efekty). Základní princip jeho fungování je následující:

1. Uživateli je nastavena cookie s utajenou hodnotou.

[23] `https://summernote.org/`
[24] `https://pypi.org/project/django-bleach/`

2. Do zobrazovaných formulářů je vloženo skryté pole, které obsahuje tuto hodnotu.
3. Pokud požadavek na metodu POST neobsahuje cookie shodující se s hodnotou v tomto poli, je vyhodnocen jako nelegitimní.

Detailní popis fungování CSRF ochrany poskytuje dokumentace frameworku Django [54].

3.5.3 Ochrana proti SQL injection

SQL injection je útok, při kterém útočník dokáže v systému spustit vlastní SQL kód, čímž může například přečíst z databáze data, ke kterým by neměl mít přístup, nebo v ní provádět neoprávněné změny.

ORM frameworku Django těmto útokům brání tak, že automaticky ošetřuje veškeré vstupy, které jsou používány při generování SQL příkazů, aby nemohlo dojít ke spuštění cizího SQL kódu.

ORM umožňuje vývojáři i použití vlastního SQL (nikoliv generovaného frameworkem). Dokumentace nicméně upozorňuje na riziko nesprávného ošetření parametrů používaných při tvorbě vlastního SQL a z toho vyplývající zranitelnost systému vůči SQL injection.

V systému SOS jsou všechny SQL dotazy generovány čistě prostřednictvím ORM, žádné riziko SQL injection tedy v systému nehrozí.

3.5.4 Ochrana proti Clickjackingu

Clickjacking je útok, při kterém útočník vloží stránku jež je cílem útoku do rámce `<iframe>` na své škodlivé stránce. Takto může oklamat uživatele a způsobit to, že provede nevědomě provede nějakou akci na stránce, která je cílem útoku.

Django poskytuje ochranu před tímto typem útoku v podobě `XFrameOptionsMiddleware`. Ten s využitím HTTP hlavičky `X-Frame-Options` omezuje možnost vkládat stránky do rámců na jiných webech. Detailní fungování této ochrany je opět popsáno v oficiální dokumentaci [55].

Systém SOS používá nastavení `X_FRAME_OPTIONS = "DENY"`, čímž je znemožněno vkládání stránek systému SOS do rámců ve všech případech.

3.5.5 Používání HTTPS

HTTPS je bezpečná verze protokolu HTTP používající protokol SSL nebo novější TLS pro šifrování komunikace mezi serverem a klientem. Tím je zabráněno odposlouchávání a narušování obsahu této komunikace. HTTPS je specifikováno standardem RFC 2818 [56].

Konfigurace frameworku Django zahrnuje několik položek týkajících se HTTP a HTTPS ovlivňující zpracovávání požadavků, jedná se především o následující:

- `SECURE_SSL_REDIRECT` – je-li tato hodnota nastavená na `True`, jsou automaticky všechny HTTP požadavky přesměrovány na jejich HTTPS alternativu. V produkčním prostředí systému SOS je tato funkcionalita vypnuta, neboť je toto řešeno na úrovni fakultní proxy. Ta mimo přesměrování požadavků zajišťuje kompletní HTTPS provoz a provádí automatickou správu HTTPS certifikátů [57].

- `SESSION_COOKIE_SECURE` a `CSRF_COOKIE_SECURE` – jsou-li tyto hodnoty nastavené na `True`, zajistí, že tyto cookies budou vždy odesílány pouze přes HTTPS.

- `SESSION_PROXY_SSL_HEADER` – dvojice hodnot představující název hlavičky požadavku a její hodnotu značící, že se jedná o HTTPS požadavek. Vzhledem k tomu, že přesměrování HTTP požadavků na HTTPS je řešeno fakultní proxy a provoz mezi proxy a serverem systému SOS je již pouze HTTP v rámci fakultní sítě, je tato hodnota nastavena na `("HTTP_X_FORWARDED_PROTO", "http")`.

Systém díky použité konfiguraci `SESSION_PROXY_SSL_HEADER` identifikuje všechny požadavky jako zabezpečené. V běžných případech by toto znamenalo závažné bezpečnostní riziko, v případě systému SOS je ale situace jiná, vzhledem k tomu, že čistě HTTP požadavky se do systému díky fakultní proxy nemohou za žádných okolností dostat.

Nabízí se otázka, zda-li nemůže být do systému odeslán HTTP požadavek, pohybuje-li se odesílatel v rámci fakultní sítě (nebo s využitím fakultní VPN). Bude-li požadavek odeslán na adresu `http://sos.fit.cvut.cz`, tak jako tak projde přes fakultní proxy a bude přesměrován na HTTPS. Pokud by z nějakého důvodu byl požadavek odeslán přímo na neveřejnou IP adresu v rámci fakultní sítě, proxy by byla skutečně obejit a a systém by skutečně HTTP požadavek přijal a považoval jej za zabezpečený. Tomu je ale zabráněno konfigurací `ALLOWED_HOSTS`, která obsahuje seznam povolených domén pro příchozí požadavky. Všechny příchozí požadavky na doménu, která není v seznamu, jsou automaticky odmítnuty. V produkčním prostředí se v tomto seznamu nachází pouze jedna hodnota – `sos.fit.cvut.cz`.

3.5.6 Session

Protokol HTTP je tzv. *stateless* – to znamená, že každý požadavek je obsluhován zvlášť, nezávisle na těch předchozích. V některých situacích je ale třeba stav udržovat, například pro realizaci přihlašování uživatelů nebo výše zmíněné CSRF ochrany. K těmto účelům framework Djnago využívá tzv. Session, která pomocí cookies umožňuje identifikovat uživatele odesílající požadavky. HTTP cookies jsou mechanismus, který umožňuje udržování stavových informací ve webovém prohlížeči a je detailně popsán v RFC 6265 [58]. Django Session je popsána v oficiální dokumentaci [59]. V této sekci autor poskytuje

stručný výtah z této dokumentace a uvádí zvolený způsob používání Session v systému SOS.

Správu Session zajišťuje `SessionMiddleware`, který do objektu `request`, jenž reprezentuje HTTP požadavek a s nímž pracují všechny Views, přidává atribut `session`. S tímto atributem lze pracovat stejně jako s obyčejným Python slovníkem a lze do něj ukládat data, která budou přístupná i při zpracování budoucích požadavků odeslaných stejným uživatelem.

Framework Django nabízí více možností, jak může být Session realizována. Výchozí možností je ukládání obsahu Session do tabulky v databázi. Cookies pak obsahují pouze identifikátor dané Session, nikolv samotná data. Tuto možnost autor zvolil i pro systém SOS, především z toho důvodu, že neumožňuje uživatelům žádným způsobem měnit data uložená v Session. Dalšími možnosti které dokumentace popisuje jsou kompletní ukládání Session pomocí cookies, ukládání dat do souboru nebo ukládání dat do cache systému.

3.5.7 Ověřování totožnosti uživatelů

Framework Django poskytuje autentizační systém, který bez potřeby dalších úprav umožňuje registraci uživatelů, přihlašování pomocí uživatelského jména a hesla. Tento systém obsahuje model uživatele pro jeho uložení do databáze a příslušné Views pro registraci, přihlášení, odhlášení a změnu hesla. Autentizační systém frameworku Django je popsán v oficiální dokumentaci [60].

Systém SOS nicméně potřebuje k autentizaci uživatelů využívat fakultní OAuth 2.0 autorizační server. Z implementace poskytované framework je tedy s určitými úpravami využita pouze část, která zajišťuje správu uživatelských účtů a samotné přihlašování. Namísto ověřování identity uživatele prostřednictvím hesla je k tomuto účelu používán již zmíněný autorizační server, a to způsobem, jenž je popsán v sekci 3.4.1.

3.6 Frontend

Během vývoje backendové části systému byla frontendová část vyvíjena jen v minimálním rozsahu potřebném pro otestování vyvíjených funkcionalit – uživatelské rozhraní bylo realizováno pouze jednoduchými HTML dokumenty, které zobrazovaly text a formuláře, bez jakýchkoliv grafických prvků. Jakmile byla dokončena většina funkcionalit systému, začal autor pracovat na vývoji řádného uživatelského rozhraní.

3.6.1 AdminLTE a další knihovny

Podstatné množství času autor ušetřil využitím volně dostupné šablony AdminLTE[25]. Díky tomu nebylo nutné vytvářet grafický design uživatelského

[25] `https://adminlte.io/`

rozhraní od začátku, ani jej implementovat ve formě CSS. Šablona využívá Bootstrap[26], což je CSS knihovna, která už sama o sobě obsahuje velké množství elementů uživatelského rozhraní. Šablona AdminLTE pak tyto elementy upravuje a přidává množství dalších. Šablona také využívá knihovnu jQuery[27], která zjednodušuje manipulaci s HTML dokumenty skriptovacím jazykem JavaScript.

Součástí šablony je i řada knihoven třetích stran, které implementují některé pokročilejší elementy, jako například textový editor Summernote[28], který systém SOS využívá ve formulářích tvorby a úpravy zadání projektu a kontrolního bodu, widget pro zadávání data a času Tempus Dominus[29] nebo DataTables[30], což jsou tabulky s pokročilými funkcemi jako je řazení podle sloupců, stránkování a vyhledávání, použité na stránce správy týmu a přehledu klasifikace.

Jedny z nejdůležitějších věcí, které poskytuje knihovna Bootstrap, jsou tzv. *breakpoints* a *grid system*. Ty umožňují poměrně snadnou implementaci responzivního designu uživatelského rozhraní a jsou tedy využívány napříč všemi stránkami uživatelského rozhraní systému. Jejich použití je popsáno v dokumentaci knihovny Bootstrap [61].

Šablona AdminLTE byla použita ve verzi 3.1.0-rc. V této verzi využívá knihovny Bootstrap 4.5.3 a jQuery 3.5.1. Všechny zdrojové kódy šablony a knihoven třetích stran jsou uloženy v adresáři `sos/staticfiles/lib/`.

3.6.2 Interakce uživatele s aplikací

Jak již bylo zmíněno v analytické části, systém SOS byl navržen jako vícestránková aplikace. V této sekci autor popisuje, jak je realizována interakce uživatele s aplikací ve frameworku Django.

V okamžiku, kdy uživatel přejde na nějakou adresu v rámci systému, je prohlížečem odeslán GET požadavek. Django s pomocí směrovače požadavků předá požadavek příslušnému View, který připraví potřebná data a vloží je do připravené HTML šablony. Výsledný HTML dokument je odeslán zpět do prohlížeče, který ho uživateli zobrazí. Následně uživatel kliknutím na některý z odkazů na zobrazené stránce přejde na jinou stránku, kde se stejný proces opakuje, nebo může vyplnit a odeslat formulář (pokud se na stránce nějaký nachází).

V takovém případě je odeslán POST požadavek obsahující data z vyplněného formuláře, ve většině případů na tu stejnou adresu, na které se již uživatel nachází. Požadavek je opět předán příslušnému View, který jej bude zpracovávat. Uvnitř View jsou data z formuláře validována pomocí Django formu-

[26] `https://getbootstrap.com/`
[27] `https://jquery.com/`
[28] `https://summernote.org/`
[29] `https://getdatepicker.com/`
[30] `https://datatables.net/`

láře, tedy instance třídy, která je potomkem `django.forms.Form`. V případě úspěšné validace dat jsou provedeny příslušné akce, jako například uložení nového záznamu do databáze, či úprava již existujícího. Následně je uživatel přesměrován na adresu určenou metodou `get_success_url()` ve View tím, že je mu odeslána odpověď s kódem 302, který značí dočasné přesměrování. V případě, že validace dat z nějakého důvodu selhala, je uživateli odeslána odpověď s kódem 200 obsahující stejnou stránku na které se již nachází, ovšem s již vyplněným formulářem a chybovými hláškami označujícími nevalidní data.

3.6.3 Struktura šablon

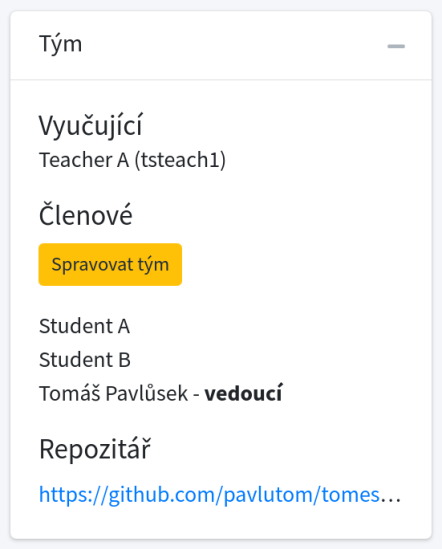
Šablonovací systém frameworku Django umožňuje využít tzv. dědičnosti mezi jednotlivými šablonami a také vkládání šablon do sebe. Syntaxi šablon včetně realizace dědičnosti a vkládání popisuje oficiální dokumentace [45]. Autor tedy vytvořil šablonu `base.html`, která obsahuje definici základní struktury HTML dokumentu, tedy tagy `<html>`, `<head>` a `<body>`. V hlavičce jsou napojeny CSS soubory AdminLTE a FontAwesome, což je CSS knihovna která obsahuje množství ikon využívaných napříč systémem. V `<body>` se nachází dva bloky – blok `site`, do kterého další šablony vkládají obsah stránky a blok `scripts`, který slouží pro vkládání JavaScript souborů. Ten již v této rodičovské šabloně obsahuje soubory knihoven AdminLTE, Bootstrap a jQuery, které tak není nutné vkládat zvlášť do všech ostatních šablon. Z šablony `base.html` dědí jednoduchá šablona přihlašovací obrazovky a šablona `app_base.html`.

Šablona `app_base.html` slouží jako základ pro obrazovky systému, které jsou zobrazovány přihlášenému uživateli. V bloku `site` této šablony jsou definovány elementy jako horní navigační lišta, postranní panel a blok `content`, do kterého další šablony vkládají obsah konkrétních stránek. V bloku `scripts` je vložen soubor `sos_base.js`, který obsahuje definice funkcí používaných v rámci celého systému, například pro realizaci volby semestru, jazyka uživatelského rozhraní nebo přepínání světlého a tmavého režimu.

V šablonách pro konktrétní stránky jsou pak vždy vyplněny bloky `title`, který slouží k definici názvu stránky zobrazovaného jako popisek karty v prohlížeči, `page_title`, což je hlavní nadpis zobrazený v horní části každé stránky a `content`, který je místem pro samotný obsah stránky. V některých šablonách dochází k rozšíření bloku `scripts` připojením dalších JavaScript souborů, buďto s definicemi dalších funkcí potřebných pro správné fungování dané stránky, nebo knihoven třetích stran, jako je například textový editor.

Ve většině šablon je (několikrát) vložena komponenta `inc/components/-card.html` (viz obrázek 3.5), která definuje vzhled karet, ze kterých se skládá uživatelské rozhraní. Této komponentě lze předat několik parametrů:

- `collapse` – určuje, jestli má být karta rozbalovací
- `collapsed` – určuje, jestli má být karta při načtení stránky sbalená



Obrázek 3.5: Komponenta karty definovaná souborem `card.html`

- `title` – nadpis karty
- `content` – cesta k souboru šablony definující obsah

```
<div class="team-card">
    {% trans "team"|capfirst as team_str %}
    {% include "inc/components/card.html" \
        with title=team_str collapse=True \
        content="teams/inc/content/team_overview.html" %}
</div>
```

Listing 8: Vložení komponenty `card.html` do šablony stránky týmu

3.6.4 Struktura uživatelského rozhraní

Při realizaci uživatelského rozhraní se autor primárně řídil vypracovaným návrhem v podobě drátěného modelu. Jak již ale bylo zmíněno, struktura jednodlivých stránek nebyla návrhem definována příliš detailně a bylo tedy potřeba přidat do stránek řadu dalších elementů. Struktura některých stránek byla v zájmu větší přehlednosti pozměněna a některé stránky byly sloučeny nebo naopak rozděleny. Byly také vytvořeny některé nové stránky, které se v původním návrhu vůbec nevyskytovaly. V této sekci autor shrnuje změny oproti původnímu návrhu, součástí čehož jsou i přepracované mapy obrazovek

odpovídající realizovanému uživatelskému rozhraní. Snímky obrazovek realizovaného uživatelského rozhraní se pak nacházejí v příloze C.

3.6.4.1 Společné prvky

Výběr semestru byl přesunut z postranního panelu do navigační lišty vedle výběru jazyka uživatelského rozhraní. Do navigační lišty také přibyl rozbalovací seznam nových notifikací, odkaz na domovskou stránku a jméno přihlášeného uživatele. Vpravo pod navigační lištou se na všech stránkách nachází tzv. *breadcrumbs*, což jsou zpětné odkazy na stránky, přes které se uživatel dostal na aktuální stránku. To je možné díky tomu, že struktura stránek je až na několik výjimek stromová.

Celé uživatelské rozhraní je dostupné v českém a anglickém jazyce, výběr jazyka se nachází v navigační liště. Uživatelské rozhraní je také možné tlačítkem v postranním panelu přepnout do tmavého režimu, což ilustruje obrázek C.6. Uživatelské rozhraní je responzivní, zobrazení na mobilním telefonu ilustruje obrázek C.18.

3.6.4.2 Přihlašovací obrazovka

Vstupním bodem do systému SOS je přihlašovací obrazovka (obrázek C.1). V produkčním prostředí obsahuje pouze jedno tlačítko „Přihlásit se“, v testovacím prostředí je zde i druhé tlačítko „Přihlásit se pomocí hesla“, které uživatele přesměruje na stránku s formulářem pro zadání uživatelského jména a hesla, což je způsob přihlášení využívaný pouze pro testovací účely. Kliknutím na tlačítko „Přihlásit se“ je zahájen proces přihlášení uživatele s využitím fakultního OAuth 2.0 autorizačního serveru, po jehož úspěšném dokončení je přihlášený uživatel přesměrován na domovskou stránku.

3.6.4.3 Seznam notifikací

Poslední položkou rozbaleného seznamu notifikací je tlačítko „Zobrazit všechna upozornění“, které uživatele přesměruje na seznam všech notifikací (obrázek C.13). Nepřečtená upozornění jsou zde označena tučným písmem. Kliknutím na položku v seznamu je uživatel přesměrován na předmět dané notifikace, což může být například ohodnocené řešení či žádost o přijetí do týmu.

3.6.4.4 Domovské stránky

Na domovské stránce studenta (obrázek C.2) se oproti původnímu návrhu nenachází samostatný přehled projektů. Místo toho se informace o vybraném projektu, jako je nadcházející termín odevzdání a stav projektu vzhledem k příštímu kontrolnímu bodu, zobrazují přímo na kartách jednotlivých předmětů. Tyto karty mají také barevný horní okraj, barva odpovídá stavu projektu:

- žlutá – ve vývoji,
- modrá – čeká na ohodnocení,
- zelená – hotovo,
- červená – vráceno.

Na domovské stránce vyučujícího (obrázek C.3) oproti návrhu chybí seznam projektů k ohodnocení, který byl pro přehlednost přesunut až na stránku konkrétního předmětu. Na kartách předmětů je zobrazena informace o počtu odevzdaných řešení v daném předmětu, které čekají na ohodnocení. Tlačítko „Vytvořit předmět“ podle návrhu přesměruje vyučujícího na stránku tvorby nového předmětu, která je zachycena na obrázku C.4.

Řešené případy užití: UC10

3.6.4.5 Stránka předmětu

Stránka předmětu z pohledu studenta (obrázek C.5) víceméně odpovídá návrhu. Na kartách projektů jsou navíc informace o složení týmu. Na kartě projektu studenta je navíc zobrazen i termín odevzdání a aktuální stav, podobně jako na domovské stránce. V horní části se v závislosti na konfiguraci předmětu mohou nacházet tlačítka „Seznam zadání“ a „Vytvořit tým“.

Stránka předmětu z pohledu vyučujícího oproti návrhu neobsahuje seznam paralelek, ale přímo seznam projektů v rámci daného předmětu a seznam odevzdaných řešení, které čekají na ohodnocení. V horní částí se nachází tlačítka, kterými lze aktivovat filtrování podle paralelky. Samostatná stránka paralelky byla z uživatelského rozhraní úplně odstraněna. Nad filtrovacími tlačítky se ještě nachází tlačítka „Přehled klasifikace“, „Nastavení předmětu“ a podle konfigurace předmětu případně „Seznam zadání“ a „Vytvořit tým“.

3.6.4.6 Konfigurace předmětu

Všichni vyučující se mohou pomocí tlačítka na stránce předmětu dostat na stárnku konfigurace. V závislosti na konfiguraci předmětu jsou jednotlivé volby buďto pouze zobrazeny (obrázek C.9), nebo je lze upravovat. V takovém případě je stránka konfigurace předmětu podobná, jako stránka se správcovským nastavením (obrázek C.10).

Na stránce se správcovským nastavením může administrátor předmětu měnit výchozí nastavení předmětu a případně povolit vytváření vlastních nastavení vyučujících. Dále zde má možnost vybrat, kteří vyučující mají být administrátory předmětu a kteří vyučující jsou zodpovědní za jednotlivé paralelky. V pravé částí obrazovky je zobrazen seznam kontrolních bodů, které je možné kliknutím na tlačítko upravit a také tlačítko pro přidání nového kontrolního bodu.

Řešené případy užití: UC11

3.6.4.7 Úpravy kontrolních bodů

V závislosti na konfiguraci definuje kontrolní body buďto administrátor předmětu globálně, nebo každý vyučující zvlášť. Stránka tvorby nového kontrolního bodu obsahuje formulář s textovým editorem a polem pro výběr výchozího termínu odevzdání, které je realizováno widgetem Tempus Dominus. Výchozí termín odevzdání lze později předefinovat pro jednotlivé paralelky. Vedle pole termínu odevzdání se nachází zaškrtávací políčko „Přepsat deadline paralelek“, jehož zaškrtnutí po uložení změn způsobí přepsání vlastních termínů odevzdání paralelek výchozím termínem.

Stránka úpravy kontrolního bodu (obrázek C.11) je stejná jako stránka tvorby, pouze navíc obsahuje v pravé části seznam paralelek a jejich termínů odevzdání. Kliknutím na tlačítko „Změnit“ je vyučující přesměrován na stránku úpravy termínu odevzdání pro konkrétní paralelku (obrázek C.12).

Řešené případy užití: UC05, UC11

3.6.4.8 Přehled klasifikace

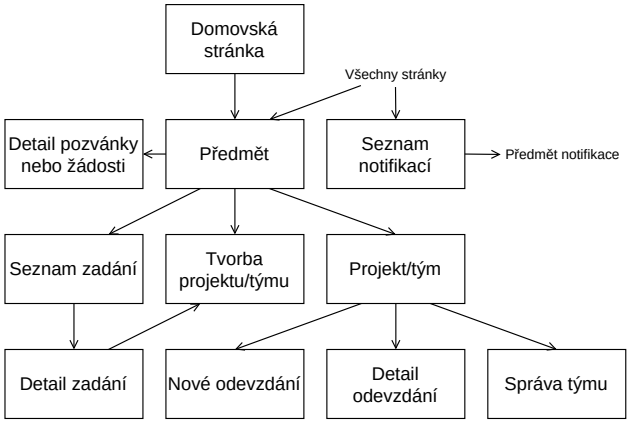
Na stránku přehledu klasifikace vyučující přeje kliknutím na tlačítko na stránce předmětu. Je zde zobrazena tabulka se všemi studenty daného předmětu a jejich bodovým hodnocením v jednotlivých kontrolních bodech a součtem bodů získaných v daném semestru. Tabulku lze seřadit podle kteréhokoliv sloupce. Pro realizaci tabulky byla použita knihovna DataTables.

Řešené případy užití: UC14

3.6.4.9 Zadání

V závislosti na konfiguraci může uživatel (student nebo vyučující) kliknutím na tlačítko na stránce předmětu přejít na stránku se seznamem zadání (obrázek C.21). Na této stránce jsou vypsána zadání, která jsou nabízena k vypracování (pokud je uživatel student) nebo která byla vytvořena vyučujícím (pokud je uživatel vyučující). Je-li uživatel vyučující, je v horní části k dispozici tlačítko pro vytvoření nového zadání a nad seznamem zadání je zobrazen formulář pro výběr výchozího zadání. Tato volba je důležitá především v případě, kdy konfigurace předmětu předepisuje společné zadání pro všechny studenty – budou vypracovávat zde zvolené zadání.

Kliknutím na položku v seznamu je uživatel přesměrován na stránku detailu zadání (obrázek C.22), na které jsou zobrazeny všechny informace týkající se daného zadání. Je-li uživatel student a umožňuje-li to konfigurace předmětu, v horní části této obrazovky je zobrazeno tlačítko „Vytvořit tým“, které studenta přesměruje na stránku tvorby týmu. Je-li uživatel vyučující, je na tomto místě zobrazeno tlačítko „Upravit“, které jej přesměruje na stránku s formulářem pro úpravu zadání (obrázek C.23). Tato stárnka je shodná se stránkou tvorby nového zadání (s výjimkou nadpisu a textu potvrzovacího tlačítka).



Obrázek 3.6: Mapa obrazovek realizované aplikace z pohledu studenta

Řešené případy užití: UC05, UC12

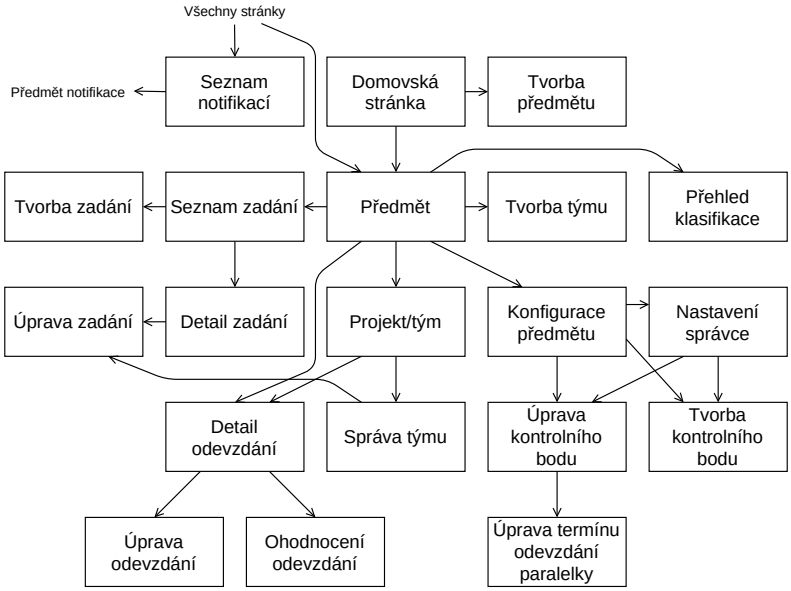
3.6.4.10 Tvorba týmu

Jedním ze způsobů, jak může student přejít na stránku tvorby týmu (obrázek C.15) je kliknutí na tlačítko na stránce předmětu. Na této stránce je zobrazen formulář, ve kterém student označí ostatní studenty, které chce do svého týmu pozvat, nastaví, zda chce povolit žádosti o přijetí do týmu a volitelně přidá odkaz na repozitář projektu. V případě, že jsou dle konfigurace předmětu projekty vypracovávány samostatně, pole týkající se ostatních členů týmu nejsou ve formuláři přítomna.

Pokud konfigurace předmětu určuje, že zadání projektů spravují vyučující, je součástí formuláře pole pro výběr z dostupných zadání vytvořených vyučujícím. V opačném případě je součástí stránky ještě druhý formulář, který slouží k tvorbě zadání a je identický s formulářem tvorby zadání pro učitele C.14). V případě, že zadání spravují vyučující, může student na stránku tvorby týmu přejít i kliknutím na tlačítko „Vytvořit tým“ na stránce detailu zadání. V takovém případě je v poli pro výběr zadání předvyplněno zadání, ze kterého se student na stránku dostal.

V případě, že dle konfigurace spravují týmy vyučující, přejde vyučující na stránku tvorby týmu kliknutím na tlačítko „Vytvořit nový tým“ na stránce předmětu. Jedná se o identickou stránku jako z pohledu studenta, který vytváří pouze tým a vybírá ze zadání vytvořených vyučujícím. Jediný rozdíl je, že po uložení formuláře se vybraní studenti stávají členy vytvořeného týmu okamžitě, namísto toho, aby jim byly odeslány pozvánky.

Řešené případy užití: UC01, UC03, UC12



Obrázek 3.7: Mapa obrazovek realizované aplikace z pohledu vyučujícího

3.6.4.11 Stránka projektu/týmu

Stránka projektu/týmu je stejná z pohledu vyučujícího i studenta. V její horní části je zobrazen popis zadání, které daný tým vypracovává. Napravo je pak karta s přehledem zůčastněných uživatelů – zodpovědný vyučující, členové týmu a testeři. Na této kartě může být zobrazeno několik tlačítek. Pokud má uživatel oprávnění spravovat tento tým (což je určeno konfigurací předmětu a tím, zda je nebo není uživatel vyučující zodpovědný za tým nebo vedoucí týmu), je zde zobrazeno tlačítko „Správa týmu“. Pokud je uživatel student, není součástí žádného týmu a tým má povolené žádosti o členství a nemá plnou kapacitu, je zde zobrazeno tlačítko „Připojit se k týmu“. Podle stejného principu zde může být také zobrazeno tlačítko „Testovat“. Pokud již student odeslal žádost o členství v týmu, je zde k dispozici tlačítko pro zrušení této žádosti. V dolní části karty je zobrazen odkaz na repozitář týmu (je-li uveden).

Dále stránka obsahuje rozbalovací karty s přehledem jednotlivých kontrolních bodů (obrázek C.17). Ty oproti původnímu návrhu nemají vlastní stránky, ale jsou součástí stránky týmu. Karty jsou ve výchozím stavu sbalené, rozbalená je ta, která patří kontrolnímu bodu, který je právě aktuální. Karta kontrolního bodu obsahuje popis kontrolního bodu, minimální a maximální počet získaných bodů a termín odevzdání. V dolní části karty se nachází menší karty představující jednotlivá odevzdání vztahující se k tomuto kontrolnímu bodu. Na nich jsou uvedeny základní informace týkající se odevzdání. Barva jejich horní části odpovídá stavu tohoto odevzdání, podle stejného principu jako barva karet předmětu na domovské stránce studenta. Součástí karty

odevzdání je tlačítko „Zobrazit“, které uživatele přesměruje na detail vybraného odevzdání. Pokud je možné k danému kontrolnímu bodu odevzdávat vypracované řešení, je v dolní části karty zobrazeno tlačítko „Nové odevzdání“.

Řešené případy užití: UC02, UC05

3.6.4.12 Správa týmu

Na stránku správy týmu (obrázek C.19) je oprávněný student nebo vyučující přesměrován kliknutím na tlačítko „Spravovat tým“ na stránce projektu/týmu. V původním návrhu tato stárnka obsahovala seznam členů týmu, seznam testerů a formulář pro pozvání nového člena týmu.

Oproti původnímu návrhu zde přibyl formulář pro pozvání testera a formulář nastavení týmu. Seznamy členů týmu a testerů byly sloučeny do jedné tabulky, ve které jsou zároveň zobrazeni i členové, kteří byli teprve pozváni, nebo odeslali žádost o přijetí do týmu. Tyto žádosti lze tlačítky v jednotlivých řádcích tabluky přijmout či zamítnout. Členům týmu lze kliknutím na tlačítko předat vedení týmu. Všem uživatelům v tabulce lze kliknutím na tlačítko „Napsat zprávu“ odeslat e-mail na fakultní e-mailovou adresu prostřednictvím výchozího e-mailového klienta uživatele.

Umožňuje-li to konfigurace předmětu, může uživatel v horní části stránky kliknutím na tlačítko „Upravit zadání“ přejít na stránku úpravy zadání, které tým vypracovává.

Řešené případy užití: UC03, UC04

3.6.4.13 Odevzdání

Kliknutím na tlačítko „Nové odevzdání“ je student přesměrován na stránku s formulářem pro odevzdání řešení (obrázek C.24). Zde je pole pro zadání poznámky a formuláře pro přidání příloh v podobě souborů nebo webových odkazů. Mezi typem přílohy lze přípínat tlačítky „Soubor“ a „URL“. Nejprve je zobrazen pouze jeden formulář přílohy, kliknutím na tlačítko „Přidat další“ lze přidat libovolný počet dalších příloh. Formulář lze uložit buďto tlačítkem „Odevzdat“, čímž dojde k odevzdání řešení vyučujícímu k ohodnocení, nebo tlačítkem „Uložit nedokončené“, kterým se odevzdání pouze uloží, například pro budoucí úpravy dalšími členy týmu.

Kliknutím na tlačítko „Zobrazit“ na kartě odevzdání na stránce projektu je uživatel přesměrován na stránku detailu tohoto odevzdání (obrázek C.25). Zde je zobrazeno datum odevzdání, jeho stav a textová poznámka studentů. Dále je k dispozici seznam příloh, které lze kliknutím zobrazit. Dále se na stránce nachází seznam testerů, kteří ještě neotestovali toto řešení a seznam testerů, jejichž testování již vedoucí týmu potvrdil. Vedoucí týmu může potvrdit otestování řešení testerem kliknutím na tlačítko „Potvrdit testování“ vedle jména testera v seznamu nepotvrzených testerů. V pravé části stránky se nachází karta s hodnocením vyučujícího. Ta nese buďto informaci o tom, že odevzdání

zatím nebylo ohodnoceno, nebo zobrazuje hodnocení, tedy textovou poznámku vyučujícího a počet získaných bodů. V případě, že je uživatel vyučující, který může hodnotit toto odevzdání, je zde zobrazeno tlačítko „Ohodnotit“ nebo „Upravit hodnocení“.

Pokud je uživatel členem týmu, v horní části obrazovky je zobrazeno tlačítko, které jej přesměruje na formulář pro úpravu tohoto odevzdání. Jedná se o stejnou stránku jako pro nové odevzdání, pouze jsou zde předvyplněny hodnoty. Existující přílohy jsou uvedeny v seznamu nad formuláři pro přidání nových příloh, spolu se zaškrtávacími políčky s popiskem „Odstranit“, jejichž zaškrtnutí povede k odstranění přílohy při uložení formuláře.

Některá odevzdání již nelze upravovat – buďto z důvodu, že již byla ohodnocena vyučujícím, nebo byla odevzdána před termínem odevzdání a aktuálně již termín odevzdání vypršel. V těchto případech je místo tlačítka pro úpravu zobrazeno tlačítko „Vytvořit kopii“, kterým může uživatel vytvořit identickou kopii tohoto odevzdání, kterou je následně možné upravit a odevzdat znovu.

Kliknutím na tlačítko „Ohodnotit“ na detailu odevzdání vyučující přejde na obrazovku hodnocení (obrázek C.26). Na té je zobrazeno pole pro zadání počtu získaných bodů s tlačítky pro usnadnění zadávání častých hodnot, pole pro textovou poznámku vyučujícího pro studenty a soukromou poznámku pro vlastní potřebu vyučujícího. V případě, že dané odevzdání proběhlo po termínu, je zde navíc pole pro zadání bodové penalizace za pozdní odevzdání.

Řešené případy užití: UC06, UC07, UC08, UC09, UC13

3.6.5 Naplnění funkčních požadavků

Výše popsané částí uživatelského rozhraní aplikace plně pokrývají jednotlivé případy užití definované návrhem, jak ukazuje tabulka 3.1. Vzhledem k tomu, že případy užití pokrývají definované funkční požadavky (jak ukazuje tabulka 2.1 v návrhu), lze dojít k závěru, že jednotlivé funkční požadavky dané návrhem byly naplněny. Jejich naplnění však bylo následně ještě ověřeno uživatelským testováním, o kterém hovoří následující kapitola.

Popsané části uživatelského rozhraní „Společné prvky“, „Přihlašovací obrazovka“ a „Seznam notifikací“ a domovská obrazovka (ve smyslu obrazovky jako takové, nikoliv popsané části uživatelského rozhraní) neřeší žádné konkrétní případy užití ani funční požadaky. Tyto části slouží k podpoře dobré orientace uživatele v uživatelském rozhraní, přihlašovací obrazovka pak naplňuje nefunkční požadavek ověřování totožnosti uživatelů.

	UC01	UC02	UC03	UC04	UC05	UC06	UC07	UC08	UC09	UC10	UC11	UC12	UC13
Společné prvky													
Přihlašovací obrazovka													
Seznam notifikací													
Domovské stránky										+			
Stránka předmětu													
Konfigurace předmětu											+		
Úpravy kontrolních bodů					+						+		
Přehled klasifikace													
Zadání					+							+	
Tvorba týmu	+		+									+	
Stránka projektu/týmu		+			+								
Správa týmu			+	+									
Odevzdání						+	+	+	+				+

Tabulka 3.1: Pokrytí případů užití částmi uživatelského rozhraní

3.7 Dokumentace

Primárním záměrem autora práce bylo vytvářet zdrojové kódy systému tak, aby nebylo pro jejich porozumění třeba rozsáhlé programátorské dokumentace. Při vývoji byl kladen důraz na správné pojmenování všech funkcí a tříd, aby byl z jejich názvů co nejlépe čitelný jejich účel. I přes to ale byla většina tříd a funkcí a také část metod jednotlivých tříd opatřena dokumentačními řetězci a komentáři, vysvětlujícími jejich specifika.

Primární dokumentací celého systému je tato bakalářská práce. Framework Django ovšem také nabízí nástroj pro automatickou tvorbu dokumentace v podobě balíčku `django.contrib.admindocs`[31]. Tato dokumentace nabízí přehled jednotlivých Modelů a jejich atributů a také přehled všech URL adres v rámci aplikace a Views, které je obsluhují. Dokumentace je vytvářena na základě metadat Modelů a dokuementačních řetězců ve zdrojových kódech. Dokumentace je přístupná uživatelům s administrátorskými oprávněními v rámci administrátorského rozhraní frameworku Django[32]. Pro snadný přístup do administrátorského rozhraní a dokumentace byly do postraního panelu uživatelského rozhraní přidány tlačítka „Administrace“ a „Dokumentace“, která jsou zobrazena uživatelům s administrátorskými oprávněními.

3.8 Zdrojové kódy

Veškeré zdrojové kódy systému SOS jsou uloženy ve webovém repozitáři GitLab, který se nachází na adrese `https://gitlab.fit.cvut.cz/pavlutom/sos`. V repozitáři se nacházejí dvě větve – `develop` a `master`. Při vývoji nové funkcionality je vytvořena nová větev, která je po dokončení vývoje sloučena s větví `develop` pomocí tzv. *Merge requestu*. V okamžiku, kdy se větev `develop` nachází ve stavu, kdy je možné a žádoucí nasadit nově přidané funkcionality do provozu, je tato větev opět pomocí *Merge requestu* sloučena s větví `master`. S každou změnou ve větvi `master` automaticky dochází k nasazení změn na server, kde je systém provozován. Tento proces je popsán v sekci 3.9.3.

Do repozitáře se zdrojovými kódy byl kromě autora práce udělen plný přístup i vedoucímu práce Ing. Jiřímu Hunkovi a oponentovi práce Ing. Janu Matouškovi. Na požádání bude přístup do repozitáře udělen i dalším osobám.

3.9 Nasazení systému do provozu

Po dokončení implementace byl systém SOS nasazen na server poskytnutý fakultou do testovacího provozu. V této sekci autor popisuje proces nasazení systému na server, realizaci CI/CD a způsob monitorování chyb při běhu systému v tomto prostředí.

[31] `https://docs.djangoproject.com/en/3.2/ref/contrib/admin/admindocs/`
[32] `https://docs.djangoproject.com/en/3.2/ref/contrib/admin/`

3.9.1 Parametry serveru

Pro nasazení systému SOS přidělilo oddělení ICT FIT ČVUT autorovi práce server – virtuální stroj na platformě CloudFIT. Stroj má následující parametry:

- **Procesor**: Intel s architekturou x86_64, maximální frekvence 3.0GHz, přidělena 2 logická jádra
- **Paměť:** 4GiB DIMM RAM
- **Pevný disk:** 50GiB SSD

Na serveru byl na požádání autora práce předinstalován operační systém GNU/Linux Debian 10 4.19.181-1 x86_64.

3.9.2 Instalace a konfigurace

Autor práce na serveru provedl konfiguraci uživatelských účtů a instalaci a konfiguraci webového serveru NGINX a databázového stroje PostgreSQL včetně vytvoření databáze pro systém SOS. Následně byly z GitLab repozitáře staženy zdrojové kódy systému SOS. Po instalaci všech potřebných knihoven a konfiguraci samotného systému SOS pro provoz na tomto konkrétním stroji byl systém bez problémů spuštěn.

Všechny provedené kroky jsou detailně popsány v instalačním manuálu v příloze A. Při instalaci systému SOS a vytváření instalačního manuálu autor práce čerpal jednak ze svých zkušeností ze zaměstnání, ale také z návodu [62] dostupného na komunitním fóru společnosti DigitalOcean, která se zabývá poskytováním virtuálních serverů.

3.9.3 Countinuous Integration / Continuous Deployment

První nasazení aplikace aplikace je součástí instalace popsané již zmíněným instalačním manuálem v příloze A. Nasazování budoucích verzí je realizováno automaticky prostřednictvím *Continuous Deployment* zajišťovaného webovým repozitářem GitLab, ve kterém se nacházejí zdrojové kódy.

V souboru `.gitlab-ci.yml` se nachází konfigurace tzv. Pipeline, což je sekvence akcí, které jsou provedny při každém nahrání nové verze do repozitáře (`git push`). Tyto akce jsou spouštěny na virtuálním stroji v rámci repozitáře GitLab. Pipeline je rozdělena do třech fází:

1. **Build:** vytvoření virtuálního prostředí a instalace potřebných Python balíčků.
2. **Test:** spuštění testů. Pokud testy selžou, Pipeline dále nepokračuje.

3. **Deploy:** nasazení nové verze. Virtuální stroj se přes SSH připojí k serveru systému SOS a vyvolá zde akce potřebné k aktualizaci na novou verzi. Jedná se o stažení nové verze kódu z repozitáře, aktualizaci Python balíčků, provedení databázových migrací a nakonec restart webové služby.

Doba přerušení provozu systému SOS při nasazování nové verze je v řádu jednotek sekund. Jedná se pouze o dobu provádění třetí fáze pipeline.

Poslední fáze Pipeline je prováděna pouze v případě, že je kód nahrán do větve `master`, v ostatních případech se spouští pouze první dvě fáze pro ověření funkčnosti dané verze.

3.9.4 Monitorování chyb

Pro monitorování chyb, které mohou v produkčním provozu nastat, systém SOS používá službu Sentry. Ta je stejně jako například GitLab dostupná v SaaS variantě a v self-hosted variantě. Fakulta informačních technologií provozuje self-hosted instanci Sentry na vlastním serveru, dostupnou na adrese `https://sentry.fit.cvut.cz`. Integrace Sentry s frameworkem Django je poměrně přímočará, zahrnuje pouze registraci ve webovém rozhraní Sentry, instalaci příslušného Python balíčku a přidání několika málo řádků kódu do konfigurace frameworku Django pro běh v produkčním prostředí. Celý postup integrace popisuje dokumentace Sentry, která je dostupná přímo ve webovém rozhraní.

Díky integraci Sentry lze mít podrobný přehled o všech neošetřených výjimkách, které byly za běhu na serveru vyvolány, včetně kontextu zahrnujícího například konkrétní místo v kódu, kde byla výjimka vyvolána či řadu o informací o připojeném klientovi – například který uživatel byl právě přihlášen nebo jaký byl použit webový prohlížeč. Na nastalé chyby dokáže Sentry také v reálném čase upozorňovat vybrané zodovědné osoby například prostřednictvím e-mailové notifikace. Tyto chyby lze později také ve webovém rozhraní Sentry seskupovat podle jejich druhu, konkrétní vyvolané výjimky či URL adresy, kde chyba nastala. Sentry také umožňuje správu chyb z hlediska jejich životního cyklu.

Kapitola 4

Testování

Testování je jednou z fází životního cyklu softwarového profuktu. V této kapitole autor práce rozebírá způsoby testování, kterým byl systém SOS podrobován. Účelem testování bylo jednak odhalení chyb, ale také ověření, že bylo dosaženo požadované funkcionality systému. Součástí kapitoly je i shrnutí problémů, které byly testováním odhaleny, spolu popisem jejich řešení či návrhem řešení.

4.1 Automatické testy

Framework Django nabízí řadu nástrojů k tvorbě automatických testů, které jsou popsány v jeho dokumentaci [63]. Doporučení, jak správně testovat Django aplikace lze nalézt také například v publikaci Two Scoops of Django [47, kap. 24].

4.1.1 Testovací data

Pro tvorbu testovacích dat použil autor knihovnu Factory Boy[33]. Ta umožňuje poměrně jednoduchým způsobem poměrně jednoduchým způsobem využít již definované Modely k vytváření testovacích dat s náhodnými, ale validními hodnotami.

Ve většině aplikací[34] systému se nachází modul `tests/`, který obsahuje soubor `factories.py`, ve kterém je využito knihovní třídy `factory.django.DjangoModelFactory` k definici příslušných Factories pro jednotlivé Modely. Factories jsou následně využívany jednak v jednotkových testech, ale také v administračních příkazech pro tvorbu testovacích dat, potřebných pro provádění uživatelského testování. Díky nim při tvorbě testovacích dat postačuje specifikovat pouze ty atributy jednotlivých modelů, na jejichž hodnotě v kon-

[33] `https://factoryboy.readthedocs.io/en/stable/`

[34] Ve smyslu Django aplikace, viz sekce 3.3.2

krétním testu záleží. Ostatní atributy jsou pak vyplněny náhodnými validními hodnotami.

4.1.2 Nástroje pro analýzu pokrytí kódu testy

Pro zjištění míry pokrytí kódu jednotkovými testy autor využil nástroj coverage.py[35]. Tento nástroj při spouštění testů dokáže rozpoznat, které řádky zdrojových kódů byly provedeny a které nikoliv, na základě čehož vypočítá míru pokrytí kódu testy. Nástroj taktéž umožňuje vygenerování podrobného reportu ve formátu HTML, který obsahuje tabulku pokrytí jednotlivých souborů se zdrojovými kódy a také samotný zdrojový kód s jednotlivými řádky barevně označenými podle toho, zda byly při spuštění testů provedeny.

Pro analýzu pokrytí kódu testy byl vytvořen jednoduchý skript `run_tests.sh`, který spustí testy spolu s nástrojem coverage.py a následně vypíše stručný report do příkazové řádky. Podrobný HTML report pak lze vygenerovat příkazem `coverage html`.

HTML report je generován i po spuštění testů v Pipeline v rámci CI/CD. Vygenerované soubory jsou uloženy v repozitáři jako tzv. artefakty[36], které jsou následně k dispozici ke stažení.

4.1.3 Pokrytí zdrojových kódů automatickými testy

Pro systém SOS bylo vytvořeno celkem 30 jednotkových testů, které testují aplikační část logiky nacházející se v jednotlivých Modelech. Zbývá však ještě vytvořit testy pro její zbylou část, která se nachází v jednotlivých Views a také pro formuláře a různé pomocné funkce.

Celkové pokrytí zdrojových kódů systému SOS automatickými testy je v současnosti 59 %. Tuto skutečnost autor práce považuje za nedostatek, který zatím stojí v cestě plnému produkčnímu použití systému SOS. Systém byl sice během vývoje neustále testován manuálně a také byl podroben uživatelskému testování, o kterém hovoří následující sekce, nicméně při takovém způsobu testování hraje zásadní roli lidský faktor a existuje zde riziko, že některé problémy zůstaly skryty.

Vytvoření kvalitnějších a rozsáhlejších automatických testů je také důležité pro další pokračování v rozvoji systému SOS. Tyto automatické testy pak bude možné využívat i jako testy regresní, tedy k průběžnému ověřování, že nebylo dalším vývojem systému a přidáváním nových funkcionalit narušeno správné fungování těch stávajících.

[35] `https://coverage.readthedocs.io/en/coverage-5.5/`
[36] `https://docs.gitlab.com/ee/ci/pipelines/job_artifacts.html`

4.2 Uživatelské testování

Po dokončení implementace systému SOS bylo realizováno uživatelské testování. Jeho účelem bylo především ověření splnění jednotlivých funkčních požadavků, ale také ověření kvality uživatelského rozhraní systému, především z hlediska přehlednosti a použitelnosti. Uživatelské testování bylo realizováno řízeným průchodem systémem podle sepsaných scénářů. Z testování byl pořízen audiovizuální záznam zachycující obrazovku testujícího subjektu a slovní komunikaci testujícího subjektu a autora práce.

4.2.1 Výběr testujících subjektů

Pro uživatelské testování byly vybrány testující subjekty z řad aktuálních a bývalých studentů Fakulty informačních technologií ČVUT v Praze. Vzhledem k tomu, že systém SOS má za cíl být co nejvíce univerzální, byly vybrány testující subjekty napříč různými ročníky a obory. Jednotlivé testující subjekty z řad studentů jsou shrnuty v následujícím seznamu:

1. testující subjekt

- 1. ročník bakalářského studia
- aktuálně pracuje na semestrální práci v předmětu BI-PA2

2. testující subjekt

- 2. ročník bakalářského studia
- vypracovával semestrální práce v předmětech BI-DBS samostatně a BI-KOM ve dvojici
- aktuálně má zapsaný předmět BI-SP1

3. testující subjekt

- 4. ročník (prodlouženého) bakalářského studia
- obor Počítačová grafika
- vypracovával semestrální práce v předmětech BI-DBS a BI-PA2 samostatně a BI-PGR v týmu
- absolvoval předměty BI-SP1 a BI-SP2

4. testující subjekt

- úspěšně dokončila bakalářské studium
- obor Informační bezpečnost
- vypracovávala semestrální práci v předmětu BI-DBS

- absolvovala předmět BI-SP1

5. testující subjekt

- 2. ročník magisterského studia
- obor Webové inženýrství
- absolvoval předmět MI-NUR, má zkušenosti se systémem NURIS
- vypracovával samostatně řadu semestrálních prací během magisterského studia

6. testující subjekt

- vyloučen v 2. ročníku bakalářského studia
- vypracovával semestrální práce v předmětech BI-DBS a BI-PA2

Z řad vyučujících bylo uživatelské testování provedeno jedním testujícím subjektem. Jednalo se o vyučujícího předmětu BI-PA2: Programování a algoritmizace 2, kterého systém SOS zaujal pro jeho potenciální možnost využití při výuce tohoto předmětu. V BI-PA2 studenti samostatně vypracovávají semestrální práce, které následně odevzdávají do systému ProgTest. Ten ale primárně slouží k automatickému vyhodnocování programovacích úloh a neumožňuje například pohodlnou správu žádostí o tzv. *code review*[37]. Aktuálně tedy studenti tyto žádosti zasílají e-mailem.

4.3 Testovací scénáře

Pro účely uživatelského testování systému SOS autor práce sestavil celkem čtyři testovací scénáře – dva z nich testují systém z pohledu studenta a dva z pohledu vyučujícího. Dohromady tyto scénáře pokrývají vešekeré funkční požadavky specifikované v analytické části této práce.

- **TS-S1 – Průchod předmětem typu MI-NUR v roli studenta:** scénář testuje průchod studenta předmětem typu MI-NUR, což znamená především kompletní tvorbu týmu studenty. Následně je testováno odevzdávání řešení a zobrazování hodnocení, které se příliš neliší od ostatních typů předmětů. Nakonec je testována základní použitelnost systému z mobilního telefonu. *Testované funkční požadavky: FP3, FP4, FP5, FP6.*
- **TS-S2 – Tvorba týmů v dalších typech předmětů:** scénář testuje proces tvorby týmů z pohledu studenta na začátku semestru pro různé konfigurace předmětu. *Testované funkční požadavky: FP3, FP4.*

[37] Při *code review* vyučující hodnotí kvalitu zdrojových kódu studenta.

- **TS-V1 – Průchod předmětem typu MI-NUR v roli vyučujícího:** scénář testuje průchod vyučujícího předmětem typu MI-NUR, což zahrnuje konfiguraci předmětu a následně ohodnocování odevzdaných řešení studentů, které se od ostatních typů předmětů příliš neliší. *Testované funkční požadavky: FP1, FP3, FP7.*
- **TS-V2 – Počáteční konfigurace dalších typů předmětů:** scénář testuje tvorbu profilů předmětů a jejich následnou konfiguraci pro různé typy předmětů, včetně správy vyučujících. *Testované funkční požadavky: FP1, FP2, FP3, FP4.*

Testování bylo navrženo tak, aby bylo možné pro urychlení celého procesu spustit studentské scénáře TS-S1 a TS-S2 (resp. učitelské scénáře TS-V1 a TS-V2) ihned po sobě. Nebylo tedy nutné mezi scénáři prováděnými jedním testujícím subjektem systém nijak konfigurovat, ani vkládat do databáze žádná další data.

Scénáře mimo jiné testují interakci testujícího subjektu s dalšími uživateli v prostředí systému SOS. Aby bylo zabráněno zvýšené náročnosti celého procesu vyplývající z koordinace více testujících subjektů, rozhodl se autor práce, že bude systém testovat v jeden okamžik vždy pouze s jedním testujícím subjektem a aktivitu ostatních uživatelů systému bude simulovat pomocí administrátorského rozhraní, do kterého mimo jiné i pro tyto účely zabudoval funkcionalitu umožňující administrátorovi vydávat se v rámci systému za ostatní uživatele.

Popis jednotlivých kroků testovacích scénářů, akcí které během testování provádí administrátor v rolích dalších uživatelů a testovacích dat potřebných k provedení scénářů se nachází v příloze D.

4.4 Proces uživatelského testování

Testující subjekt by měl mít v ideálním případě k dispozici fakultní účet, kterým se dokáže autorizovat prostřednictvím fakultního OAuth 2.0 serveru. Z pohledu uživatele identický způsob autorizace využívá například systém FIT Klasifikace. Pokud k takovému účtu testující subjekt přístup neměl, využil varianty přihlášení pomocí uživatelského jména a hesla. Tento způsob přihlašování systém SOS v zájmu bezpečnosti umožňuje pouze v testovacím režimu. Pro využití této varianty bylo nutné v administrátorském rozhraní vytvořit příslušný uživatelský účet, jelikož takto nemůže být vytvořen automaticky z dat získaných z KOSapi.

Pro spuštění potřebných administračních příkazů během testování využíval autor práce vzdálený přístup k serveru přes SSH. Pro každý testující subjekt byla databáze systému SOS uvedena do iniciálního stavu, kdy jsou pouze vytvořeny příslušné tabulky bez dat. Následně byla spuštěním administračního příkazu vytvořena testovací data. Součástí testovacích dat byl také

administrátorský účet, kterým se do systému přihlásil autor práce, aby mohl pro testující subjekt simulovat akce ostatních uživatelů.

Jakmile se testovací subjekt poprvé přihlásil do systému, čímž byl vytvořen jeho uživatelský účet, spustil autor práce na serveru další administrační příkaz, kterým byl tento uživatelský účet propojen s testovacími daty. Tímto propojením jsou myšleny akce, jako například nastavení uživatele jako vyučujícího či studenta daného předmětu, či přiřazení do konkrétní paralelky.

Výše zmíněné administrační příkazy jsou spolu s jednotlivými kroky testovacích scénářů popsány v příloze D.

4.5 Výsledky uživatelského testování

Po otestování systému podle testovacích scénářů autor práce provedl s testujícími subjekty krátký rozhovor. Dotazy směřovaly na jejich celkový dojem ze systému, intuitivnost a přehlednost uživatelského rozhraní, jimi vypozorované nedostatky systému a jejich náměty na zlepšení.

4.5.1 Studenti

Testující subjekty uživatelské rozhraní systému hodnotili jako přehledné, dokázali se v něm bez větších problémů zorientovat. Nalezli zde všechny funkcionality, které od systému pro odevzdávání semestrálních prací očekávali a které bylo třeba využít pro splnění testovacích scénářů, s výjimkou celkového přehledu získaných bodů ve vybraném předmětu. O tomto problému autor práce věděl, ale bohužel jej nestihl vyřešit před zahájením uživatelského testování. Subjekty také ocenili přítomnost systému notifikací a možnost přepnutí uživatelského rozhraní do tmavého režimu.

Subjekty často systém SOS porovnávali se systémem Moodle, který je běžně využíván k odevzdávání projektů v řadě předmětů vyučovaných na fakultě. Subjekty ocenili jednoduchost uživatelského rozhraní systému SOS, které v porovnání se systémem Moodle neobsahuje nevyužité a dle jejich slov zbytečné prvky, které by negativně ovlivňovaly jejich schopnost orientace v systému.

4.5.2 Vyučující

Testující subjekt v uživatelském rozhraní samostatně orientoval a rovněž bez vetších problému dokončil oba testovací scénáře. Testující subjekt se navíc neřídil pouze sepsaným scénářem, ale projevil i vlastní iniciativu při zkoumání reakcí systému na některé nestandartní situace. To vedlo k odhalení několika chyb, které bylo následně potřeba opravit. Testující subjekt rovněž poskytl řadu návrhů na zlepšení použitelnosti systému.

4.6 Vyhodnocení zjištěných nedostatků a provedené změny

Z rozhovorů s testujícími subjekty, pozorování subjektů během testování a analýzy pořízených audiovizuálních záznamů vyplynula řada méně či více závažných nedostatků či námětů na zlepšení. Celkem bylo zaznamenáno 63 položek, u kterých byla následně zhodnocena jejich závažnost a náročnost implementace řešení. Celkem byly rozlišeny tři stupně závažnosti:

1. Nízká – nemá žádný nebo zanedbatelný vliv na použitelnost aplikace
2. Střední – může mít vliv na použitelnost aplikace, problém by měl být odstraněn
3. Vysoká – má kritický dopad na použitelnost aplikace, problém musí být odstraněn

Tabulka 4.1 uvádí počty nalezených a vyřešených nedostatků podle stupně jejich závažnosti. Více než polovina všech nedostatků byla vyřešena a až na dvě výjimky mají všechny nevyřešené nízkou závažnost. Některé nedostatky nebylo možné vyřešit v rámci této práce, buďto z časových důvodů, nebo proto, že nebylo jasné, jak a zdali vůbec by měl být uvedený problém řešen.

4.6.1 Příklady zjištěných nedostatků

Jedním z nejvýraznějších nedostatků byla absence přehledného souhrnu získaných bodů v uživetelském rozhraní studenta. O tomto nedostatku autor práce věděl i před zahájením uživatelského testování, nicméně jej nestihl včas napravit. Po dokončení uživatelského testování byl tento nedostatek opraven spolu s řadou dalších.

Další problémy se týkaly neintuitivního ovládání některých widgetů, především toho, který slouží pro výběr několika studentů při tvorbé týmu. Toto bylo vyřešeno výměnou výchozího widgetu za pokročilejší widget Select2[38]. Dalším problematickým prvkem se ukázala být tabulka s přehledem klasifikace všech studentů, které chyběly funkcionality stránkování a filtrování, což zpomalovalo orientaci vyučujícího především při vysokém počtu studentů. Tyto funkcionality byly následně k tabulce přidány.

Jedním ze závažných problémů byl nesprávně fungující import dat z KOSu pro některé předměty. Ukázalo se, že systém není schopný kompletně importovat paralelky s vysokým počtem studentů. Také bylo zjištěno, že je potřeba umožnit volbu typu paralelek, které mají být naimportovány, nikolv vždy importovat pouze cvičení. Tento problém byl následně také vyřešen, a to úpravou kódu zajišťujícího realizujícího dotazy na KOSapi a přidáním možnosti výběru typu paralelek do formuláře volby předmětu pro import.

[38] https://select2.org/

Závažnost	Celkový počet	Počet vyřešených	Počet nevyřešených
Nízká	47	18	29
Střední	13	11	2
Vysoká	4	4	0
Celkem	64	33	31

Tabulka 4.1: Shrnutí zjištěných a vyřešených nedostatků

Další připomínky testujících subjektů se týkaly především matoucích textů různých tlačítek a odkazů napříč systémem, což bylo vyřešeno jejich přetextováním, přidáním ikon a lepším barevným rozlišením. Také bylo objeveno několik prvků uživatelského rozhraní, které se zobrazovaly nesprávně v důsledku programátorských chyb. Tyto chyby byly rovněž opraveny.

Detailní přehled všech objevených nedostatků, zhodnocení jejich závažnosti a náročnosti řešení, informace zda byly vyřešeny a případně popis jejich řešení, nebo návrh řešení v případě, že vyřešeny nebyly, se nachází v příloze E. V případě, že problémy vyřešeny nebyly i z jiného nežli časového důvodu, obsahuje návrh řešení i vysvětlení tohoto důvodu.

4.6.2 Shrnutí

Všem testujícím subjektům se bez větších problémů podařilo dokončit testovací scénáře. Testovací scénáře dohromady pokrývají veškeré specifikované funkční požadavky, lze tedy usoudit, že funkční požadavky systém SOS naplňuje.

Odhalené nedostatky systému a návrhy na zlepšení poskytnuté testujícími subjekty byly zaznamenány a všechny ty, které zásadním způsobem ovlivňují použitelnost systému, byly opraveny. Opravena byla také řada drobných nedostatků zhoršujících přehlednost uživatelského rozhraní. Některé problémy se nepodařilo vyřešit v rámci této práce, nejedná se ale o problémy bránící využití systému SOS v zamýšleném rozsahu.

Všechny odhalené nedostatky a návrhy na zlepšení, které nebyly v rámci této práce řešeny, byly zaznamenány do repozitáře ve službě GitLab v sekci Issues. Spolu s nimi zde bylo zaznamenáno i několik dalších drobných nedostatků, které se při uživatelském testování neprojevily, ale autor práce si jich je vědom. Všechny tyto položky budou součástí dalšího vývoje projektu.

Závěr

Cílem této bakalářské práce bylo vyvinout SOS – Studentský odevzdávací systém, což je informační systém sloužící k odevzdávání a hodnocení týmových i individuálních semestrálních prací v předmětech vyučovaných na Fakultě informačních technologií Českého vysokého učení technického v Praze. To obnášelo provedení návrhu, implementaci všech částí systému a nasazení systému na fakultní server.

Při analýze potřeb studentů a vyučujících bylo využito poznatků z diplomové práce, jež se zabývala vývojem systému NURIS – informačního systému pro správu předmětu MI-NUR: Návrh uživatelského rozhraní. Byla také provedena analýza potřeb dalších předmětů vyučovaných na Fakultě informačních technologií, ve kterých studenti rovněž vypracovávají semestrální práce. Byly analyzovány funkční i nefunkční požadavky a jednotlivé uživatelské role, na základě čehož byl sestaven model případů užití.

Na základě této analýzy byla navržena vhodná architektura systému, systém SOS byl navržen jako vícestránková webová aplikace využívající relační databázi pro perzistenci dat. Bylo navrženo uživatelské rozhraní systému v podobě drátěných modelů a jejich vazeb na specifikované případy užití. Návrh se dále zabýval napojením systému na další informační systémy využívané na Fakultě informačních technologií. Součástí návrhu byla i volba vhodných technologií pro implementaci všech částí systému, mimo jiné byl pro realizaci backendové i frontendové části systému zvolen framework Django. Návrh se rovněž zabýval využitím systémů pro správu verzí a CI/CD.

Následně bylo přistoupeno k samotné implementaci všech částí systému SOS. Nejprve byla implementována backendová část systému, tedy datový model a potřebná aplikační logika. To zahrnovalo i implementaci funkcionality importování dat ze systému KOS a ověřování totožnosti uživatelů prostřednictvím fakultního OAuth 2.0 autorizačnícho serveru. Následně pak byla implementována část frontendová, reálné uživatelské rozhraní bylo oproti původnímu návrhu mírně upraveno a rozšířeno o další obrazovky. K vytvoření uživatelsky přívětivého responzivního uživatelského rozhraní bylo využito ša-

blony AdminLTE postavené na CSS knihovně Bootstrap. Při vývoji celého systému bylo také dbáno na jeho dostatečné zabezpečení.

Po dokončení implementace byl systém SOS nasazen na virtuální server na platformě CloudFIT. Součástí nasazení byla realizace CI/CD pomocí webového repozitáře GitLab a také využití služby Sentry pro monitorování případných problémů.

Nakonec bylo provedeno uživatelské testování, kterého se zůčastnili jak studenti, tak i vyučující. Uživatelským testováním bylo ověřeno naplnění specifikovaných funkčních požadavků, ale zároveň z něj také vyplynulo několik více či méně závažných nedostatků. Ty závažnější byly následně opraveny, stejně jako část těch méně závažných. Nedostatky nebo návrhy na zlepšení, které nebyly v rámci této práce řešeny, neovlivňují použitelnost systému v rozsahu ve kterém byl navržen.

Výsledkem bakalářské práce je funkční software splňující všechny specifikované požadavky. Na základě všech výše uvedených skutečností lze cíl práce považovat za splněný.

Možnosti budoucího rozvoje projektu

Pro plné produkční využití systému je potřeba jeho důkladnější otestování. To znamená jednak zajistit kvalitnější pokrytí zdrojových kódů automatickými testy, což zajistí dostatečnou stabilitu systému pro produkční provoz a také usnadní jeho další vývoj. Současně by to mělo obnášet také nasazení systému do testovacího provozu při výuce, například v předmětu MI-NUR: Návrh uživatelského rozhraní v následujícím semestru, čímž bude získána rozsáhlejší zpětná vazba od skutečných uživatelů jak z řad studentů, tak i vyučujících, mezi které patří především vedoucí této práce Ing. Jiří Hunka.

K získání rozsáhlejší zpětné vazby od studentů a vyučujících předmětu bude možné využít dotazníků a rozhovorů s reálnými uživateli systému, ale také například strojové analýzy chování uživatelů v systému pomocí tzv. *heatmap*. K tomu lze využít například nástroje jako Hotjar nebo Mouseflow.

Součástí dalšího vývoje systému by také mělo být získání zpětné vazby od vyučujících co nejvíce různých předmětů a zapracování jejich návrhů a požadavků. Již během uživatelského testování bylo zaznamenáno několik takových požadavků od vyučujícího předmětu BI-PA2: Programování a algoritmizace 2. Ty se týkaly například lepší optimalizace uživatelského rozhraní pro předměty s velkým množstvím projektů či pokročilejší správy zadání semestrálních prací.

Dalším možným směrem rozvoje projektu je integrace systému s dalšími informačními systémy využívanými při výuce na Fakultě informačních technologií. Především by bylo vhodné SOS propojit se systémem FIT Klasifikace, což by jednak významně usnadnilo práci vyučujícím, kteří by nebyli nuceni manuálně přenášet bodová hodnocení studentů z SOS do FIT Klasifikace, a

zároveň by díky propojení bylo možné v systému SOS zobrazovat i další položky klasifikace studentů, nikoliv pouze semestrální práci.

Dalším systémem, se kterým by bylo užitečné SOS propojit, je například platforma Courses, využívaná pro publikování studijních materiálů včetně zadání semestrálních projektů. V případě, že v daném předmětu definuje zadání vyučující, by mohla být zadání nacházající se v Courses zobrazována v SOS.

Na dalším vývoji projektu mohou v budoucnu pracovat například studenti Fakulty informačních technologií v rámci týmových projektů v předmětech BI-SP1: Softwarový týmový projekt 1 a BI-SP2: Softwarový týmový projekt 2. Především by se ale na dalším vývoji systému SOS rád podílel autor této práce v rámci navazujícího magisterského studia na Fakultě informačních technoligií, bude-li mu to umožněno.

Bibliografie

1. JIROUT, David. *Informační systém pro MI-NUR*. Praha, 2019. Diplomová práce. České vysoké učení technické v Praze, Fakulta informačních technologií.
2. HUNKA, Jiří. *Návrh uživatelského rozhraní: Hodnocení a semestrální práce / The semestral work classification* [online]. 2020 [cit. 2021-05-04]. Dostupné z: `https://courses.fit.cvut.cz/MI-NUR/classification/index.html`.
3. KŘIKAVA, Filip. *Object-Oriented Programming: Projects* [online]. 2020 [cit. 2021-05-04]. Dostupné z: `https://courses.fit.cvut.cz/BI-OOP/projects/index.html`.
4. KŘIKAVA, Filip. *Object-Oriented Programming: Project* [online]. 2019 [cit. 2021-05-04]. Dostupné z: `https://courses.fit.cvut.cz/BI-OOP/@B191/project/index.html`.
5. SUCHÁNEK, Marek. *Konceptuální modelování: Semestrální práce* [online]. 2020 [cit. 2021-05-04]. Dostupné z: `https://courses.fit.cvut.cz/BI-KOM/semestral.html`.
6. *B202-BI-SP1,BIK-SP1 - Softwarový týmový projekt 1* [online]. 2020 [cit. 2021-05-04]. Dostupné z: `https://moodle-vyuka.cvut.cz/course/view.php?id=5075`.
7. *B201-BI-SP2.1,BIK-SP2.1 - Softwarový týmový projekt 2* [online]. 2020 [cit. 2021-05-04]. Dostupné z: `https://moodle-vyuka.cvut.cz/course/view.php?id=3913`.
8. IEEE Recommended Practice for Software Requirements Specifications. *IEEE Std 830-1998*. 1998, s. 1–40. Dostupné z DOI: `10.1109/IEEESTD.1998.88286`.

9. EELES, Peter. *Capturing Architectural Requirements* [online]. 2005 [cit. 2021-05-04]. Dostupné z: https://web.archive.org/web/20201112020231/http://www.ibm.com/developerworks/rational/library/4706.html.

10. LARMAN, Craig. *Applying UML and patterns: an introduction to object-oriented analysis and design and iterative development*. 3rd ed. Upper Saddle River: Prentice Hall PTR, 2004. ISBN 01-314-8906-2.

11. *Web Application* [online]. 2014 [cit. 2021-05-04]. Dostupné z: https://techterms.com/definition/web_application.

12. SOLOVEI, V.; OLSHEVSKA, Olga; BORTSOVA, Y. THE DIFFERENCE BETWEEN DEVELOPING SINGLE PAGE APPLICATION AND TRADITIONAL WEB APPLICATION BASED ON MECHATRONICS ROBOT LABORATORY ONAFT APPLICATION. - . 2018, roč. 10. Dostupné z DOI: 10.15673/atbp.v10i1.874.

13. BUSCHMANN, Frank; MEUNIER, Regine; ROHNERT, Hans; SOMMERLAD, Peter; STAL, Michael. *Pattern-oriented software architecture: a system of patterns*. Chichester: Wiley, 1998. ISBN 04-719-5869-7.

14. BERNARD, Borek. *Prezentační vzory z rodiny MVC* [online]. 2009 [cit. 2021-05-04]. Dostupné z: https://zdrojak.cz/clanky/prezentacni-vzory-zrodiny-mvc/.

15. HOLOVATY, Adrian; KAPLAN-MOSS, Jacob. *The definitive guide to Django: Web development done right*. 2nd ed. Berkeley: Apress, c2009. ISBN 978-1-4302-1936-1.

16. GLOBALSTATS. *Desktop vs Mobile vs Tablet Market Share Worldwide* [online]. 2021 [cit. 2021-05-04]. Dostupné z: https://gs.statcounter.com/platform-market-share/desktop-mobile-tablet/worldwide/.

17. MARCOTTE, Ethan. *Responsive Web Design* [online]. 2010 [cit. 2021-05-04]. Dostupné z: https://alistapart.com/article/responsive-web-design/.

18. GOOGLE. *Mobile-first indexing best practices* [online]. 2021 [cit. 2021-05-11]. Dostupné z: https://developers.google.com/search/mobile-sites/mobile-first-indexing.

19. BROWN, Daniel M. *Communicating design: Developing web site documentation for design and planning*. Upper Saddle River, NJ: New Riders Publishing, 2006.

20. SOLID IT. *DB-Engines Ranking* [online]. 2021 [cit. 2021-05-04]. Dostupné z: https://db-engines.com/en/ranking.

21. DJANGO SOFTWARE FOUNDATION. *Databases* [online]. 2021. Verze 3.2 [cit. 2021-05-05]. Dostupné z: https://docs.djangoproject.com/en/3.2/ref/databases/.

22. HAAG, Jonas; PREMOLI, Flavio Percoco; RUSZCZEWSKI, Wojtek. *Django MongoDB Engine: MongoDB backend for Django* [online]. 2017 [cit. 2021-05-05]. Dostupné z: `https://django-mongodb-engine.readthedocs.io/`.

23. MOZILLA FOUNDATION. *What is a web server?* [online]. 2021 [cit. 2021-05-04]. Dostupné z: `https://developer.mozilla.org/en-US/docs/Learn/Common_questions/What_is_a_web_server`.

24. EBY, P.J. *Python Web Server Gateway Interface v1.0.1*. 2010. Dostupné také z: `https://www.python.org/dev/peps/pep-3333/`. PEP.

25. *Single Sign On (jednotné přihlašování)* [online]. 2019 [cit. 2021-05-04]. Dostupné z: `https://ist.cvut.cz/nase-sluzby/single-sign-on/`.

26. JIRŮTKA, Jakub. *OAuth 2.0* [online]. 2017 [cit. 2021-05-04]. Dostupné z: `https://rozvoj.fit.cvut.cz/Main/oauth2`.

27. HARDT, D. *The OAuth 2.0 Authorization Framework*. 2012. Dostupné také z: `http://tools.ietf.org/rfc/rfc6749.txt`. RFC. IETF.

28. JIRŮTKA, Jakub. *KOSapi* [online]. 2015 [cit. 2021-05-04]. Dostupné z: `https://kosapi.fit.cvut.cz/projects/kosapi/wiki`.

29. KUČERA, František. *Distribuované verzovací systémy* [online]. 2011 [cit. 2021-05-04]. Dostupné z: `https://www.abclinuxu.cz/clanky/distribuovane-verzovaci-systemy-uvod-1`.

30. FOWLER, Martin. Continuous Integration [online]. 2006 [cit. 2021-05-04]. Dostupné z: `https://martinfowler.com/articles/continuousIntegration.html`.

31. SHAHIN, Mojtaba; BABAR, Muhammad Ali; ZAHEDI, Mansooreh; ZHU, Liming. Beyond Continuous Delivery: An Empirical Investigation of Continuous Deployment Challenges. In: *2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. 2017, s. 111–120. Dostupné z DOI: `10.1109/ESEM.2017.18`.

32. DJANGO SOFTWARE FOUNDATION. *django.contrib.postgres* [online]. 2021. Verze 3.2 [cit. 2021-05-05]. Dostupné z: `https://docs.djangoproject.com/en/3.2/ref/contrib/postgres/`.

33. W3TECHS. *Comparison of the usage statistics of Nginx vs. Apache vs. Microsoft-IIS for websites* [online]. 2021 [cit. 2021-05-05]. Dostupné z: `https://w3techs.com/technologies/comparison/ws-apache,ws-microsoftiis,ws-nginx`.

34. DREAMHOST. Web server performance comparison [online]. 2021 [cit. 2021-05-05]. Dostupné z: `https://help.dreamhost.com/hc/en-us/articles/215945987-Web-server-performance-comparison`.

35. DJANGO SOFTWARE FOUNDATION. *How to use Django with Gunicorn* [online]. 2021. Verze 3.2 [cit. 2021-05-05]. Dostupné z: https://docs.djangoproject.com/en/3.2/howto/deployment/wsgi/gunicorn/.

36. GITLAB. *GitLab Documentation* [online]. 2021 [cit. 2021-05-05]. Dostupné z: https://docs.gitlab.com/.

37. PYTHON SOFTWARE FOUNDATION. *Virtual Environments and Packages* [online]. 2021. Verze 3.9.5 [cit. 2021-05-05]. Dostupné z: https://docs.python.org/3/tutorial/venv.html.

38. DJANGO SOFTWARE FOUNDATION. *Models* [online]. 2021. Verze 3.2 [cit. 2021-05-05]. Dostupné z: https://docs.djangoproject.com/en/3.2/topics/db/models/.

39. DJANGO SOFTWARE FOUNDATION. *QuerySet API reference* [online]. 2021. Verze 3.2 [cit. 2021-05-05]. Dostupné z: https://docs.djangoproject.com/en/3.2/ref/models/querysets/.

40. DJANGO SOFTWARE FOUNDATION. *Migrations* [online]. 2021. Verze 3.2 [cit. 2021-05-05]. Dostupné z: https://docs.djangoproject.com/en/3.2/topics/migrations/.

41. DJANGO SOFTWARE FOUNDATION. *URL dispatcher* [online]. 2021. Verze 3.2 [cit. 2021-05-05]. Dostupné z: https://docs.djangoproject.com/en/3.2/topics/http/urls/.

42. DJANGO SOFTWARE FOUNDATION. *Writing views* [online]. 2021. Verze 3.2 [cit. 2021-05-05]. Dostupné z: https://docs.djangoproject.com/en/3.2/topics/http/views/.

43. DJANGO SOFTWARE FOUNDATION. *Class-based views* [online]. 2021. Verze 3.2 [cit. 2021-05-05]. Dostupné z: https://docs.djangoproject.com/en/3.2/topics/class-based-views/.

44. DJANGO SOFTWARE FOUNDATION. *Generic editing views* [online]. 2021. Verze 3.2 [cit. 2021-05-05]. Dostupné z: https://docs.djangoproject.com/en/3.2/ref/class-based-views/generic-editing/#createview.

45. DJANGO SOFTWARE FOUNDATION. *The Django template language* [online]. 2021. Verze 3.2 [cit. 2021-05-05]. Dostupné z: https://docs.djangoproject.com/en/3.2/ref/templates/language/.

46. DJANGO SOFTWARE FOUNDATION. *Forms* [online]. 2021. Verze 3.2 [cit. 2021-05-05]. Dostupné z: https://docs.djangoproject.com/en/3.2/ref/forms/.

47. FELDROY, Daniel; FELDROY, Audrey. *Two Scoops of Django 3.x: Best Practices for the Django Web Framework* [online]. Two Scoops Press, 2020. Alpha. Dostupné také z: https://www.feldroy.com/products/two-scoops-of-django-3-x.

48. DJANGO SOFTWARE FOUNDATION. *django.contrib.auth* [online]. 2021. Verze 3.2 [cit. 2021-05-13]. Dostupné z: https://docs.djangoproject.com/en/3.2/ref/contrib/auth/.

49. JIRŮTKA, Jakub. *Authorization Code Grant* [online]. GitHub, 2015 [cit. 2021-05-05]. Dostupné z: https://github.com/cvut/zuul-oaas/wiki/Authorization-Code-Grant.

50. JIRŮTKA, Jakub. *RESTful zdroje* [online]. 2015 [cit. 2021-05-05]. Dostupné z: https://kosapi.fit.cvut.cz/projects/kosapi/wiki/Resources.

51. JIRŮTKA, Jakub. *XPartial* [online]. 2015 [cit. 2021-05-05]. Dostupné z: https://kosapi.fit.cvut.cz/projects/kosapi/wiki/XPartial.

52. DJANGO SOFTWARE FOUNDATION. *Security in Django* [online]. 2021. Verze 3.2 [cit. 2021-05-06]. Dostupné z: https://docs.djangoproject.com/en/3.2/topics/security/.

53. FIELDING, R.; RESCHKE, J. *Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content*. 2014. Dostupné také z: http://tools.ietf.org/rfc/rfc7231.txt. RFC. IETF.

54. DJANGO SOFTWARE FOUNDATION. *Cross Site Request Forgery protection* [online]. 2021. Verze 3.2 [cit. 2021-05-06]. Dostupné z: https://docs.djangoproject.com/en/3.2/ref/csrf/.

55. DJANGO SOFTWARE FOUNDATION. *Clickjacking Protection* [online]. 2021. Verze 3.2 [cit. 2021-05-06]. Dostupné z: https://docs.djangoproject.com/en/3.2/ref/clickjacking/.

56. RESCORLA, E. *HTTP Over TLS*. 2000. Dostupné také z: http://tools.ietf.org/rfc/rfc2818.txt. RFC. IETF.

57. *Webový server* [online]. 2021 [cit. 2021-05-06]. Dostupné z: https://help.fit.cvut.cz/cloud-fit/examples/web-server.html.

58. BARTH, A. *HTTP State Management Mechanism*. 2011. Dostupné také z: http://tools.ietf.org/rfc/rfc6265.txt. RFC. IETF.

59. DJANGO SOFTWARE FOUNDATION. *How to use sessions* [online]. 2021. Verze 3.2 [cit. 2021-05-06]. Dostupné z: https://docs.djangoproject.com/en/3.2/topics/http/sessions/.

60. DJANGO SOFTWARE FOUNDATION. *User authentication in Django* [online]. 2021. Verze 3.2 [cit. 2021-05-06]. Dostupné z: https://docs.djangoproject.com/en/3.2/topics/auth/.

61. BOOTSTRAP TEAM. *Overview* [online]. 2021. Verze 4.6 [cit. 2021-05-06]. Dostupné z: https://getbootstrap.com/docs/4.6/layout/overview/.

62. ELLINGWOOD, Justin; JETHA, Hanif. *How To Set Up Django with Postgres, Nginx, and Gunicorn on Debian 10* [online]. DigitalOcean, 2019 [cit. 2021-05-05]. Dostupné z: `https://www.digitalocean.com/community/tutorials/how-to-set-up-django-with-postgres-nginx-and-gunicorn-on-debian-10`.

63. DJANGO SOFTWARE FOUNDATION. *Writing and running tests* [online]. 2021. Verze 3.2 [cit. 2021-05-10]. Dostupné z: `https://docs.djangoproject.com/en/3.2/topics/testing/overview/`.

Příloha A

Instalační manuál

Instalační manuál popisuje postup kompletního zprovoznění systému SOS na serveru, na který je přistupováno přes SSH. Tento manuál předpokládá, že je na serveru nainstalován operační systém GNU/Linux Debian 10, nicméně po přizpůsobení některých kroků (zejména instalace balíčků třetích stran) lze manuál použít i pro jiné linuxové distribuce. Pro instalaci je nutné mít buďto k dispozici zdrojové kódy nacházející se na přiloženém paměťovém médiu, nebo přístup do GitLab repozitáře, v němž jsou zdrojové kódy uloženy. Instalace předpokládá schopnost připojit se na cílový server přes SSH jako uživatel *root*, tedy s administrátorskými oprávněními.

Několik příkazů uvedených v manuálu, které jsou spouštěny nikoliv na serveru, ale lokálně, je určených pro operační systém Linux. Pro provedení instalace z jiného typu operačního systému je potřeba použít jejich ekvivalenty pro daný systém.

Instalační manuál popisuje postup k nasazení systému do produkčního prostředí. Pro testovancí konfiguraci je postup identický, pouze jsou hodnoty „production“ vyskytující se napříč manuálem nahrazeny hodnotou „stage“.

A.1 Vytvoření SSH klíče

Prvním krokem instalace je vytvoření SSH klíče, který bude sloužit pro komunikaci serveru s GitLab repozitářem. Pokud je instalace prováděna ze zdrojových kódu z přiloženého paměťového média a není zamýšleno využívání CI/CD, tento krok může být přeskočen. Spuštěním následujícího příkazu budou vytvořeny dva soubory obsahujicí privátní klíč a veřejný klíč:

```
ssh-keygen -t rsa -f deploy_key
```

Dále je potřeba přidat tento klíč do lokálního SSH agenta, díky čemuž bude možné pomocí mechanizmu *agent forwarding* ze serveru číst GitLab repozitář. K tomu slouží následující příkaz:

```
cat deploy_key | ssh-add -
```

Nakonec je potřeba privátní klíč uložit jako tzv. *Deploy key* do GitLab repozitáře projektu, což je popsáno v příslušné dokumentaci dostupné na adrese `https://docs.gitlab.com/ee/user/project/deploy_keys/`.

A.2 Konfigurace operačního systému

Dalším krokem je připojení se na server po uživatelem *root*:

```
ssh <adresa_serveru> -l root
```

Následuje vytvoření uživatele *deploy*:

```
useradd -m -U -s /bin/bash deploy
```

Pokud byl v minulém kroku vytvořen SSH klíč podobě souborů `deploy_key` a `deploy_key.pub`, je nyní potřeba vložit obsah souboru `deploy_key.pub` na konec souboru `/home/deploy/.ssh/authorized_keys`, případně tento soubor a adresář vytvořit, pokud neexistují. Je důležité aby vlastník souboru byl uživatel *deploy*, nikoliv *root*, zajistí například následující příkaz:

```
chown -R deploy:deploy /home/deploy/.ssh
```

Pro pozdější korektní spouštění administračních příkazů systému SOS je potřeba přidat řádky s příkazy

```
export DJANGO_SETTINGS_MODULE=conf.settings.production
export PYTHONIOENCODING="UTF-8"
```

do souboru `/home/deploy/.bashrc`.

Dalším krokem je instalace databázového stroje ProstgreSQL a vytvoření databáze. Návod k instalaci pro linuxovou distribuci Debian je dostupný na adrese `https://www.postgresql.org/download/linux/debian/`. Pro vytvoření databáze je třeba se přihlásit do terminálu databázového stoje příkazem

```
su - postgres -c psql
```

a spustit následující SQL příkazy:

```
create database sos
with encoding='UTF-8' lc_ctype='C.UTF-8' lc_collate='C.UTF-8'
template template0;
create user sos with password <heslo>;
grant all privileges on database sos to sos;
```

Po instalaci databázového stroje zbývá ještě nainstalovat a nakonfigurovat webový server NGINX. Instalaci zajistí následující příkaz:

```
apt install nginx
```

Následuje vytvoření souboru `/etc/nginx/sites-available/sos` s následujícím obsahem:

```
server {
    listen 80;
    server_name <adresa_serveru>;

    location /static/ {
        root /srv/www/sos/sos/;
    }

    location / {
        include proxy_params;
        proxy_pass http://unix:/run/gunicorn.sock;
    }
}
```

Po vyvtoření souboru je potřeba vytvořit symbolický link na tento soubor v adresáři `/etc/nginx/sites-enabled/`. Z tohoto adresáře je také potřeba odstranit soubor `default`:

```
ln -s /etc/nginx/sites-available/sos /etc/nginx/sites-enabled/sos
rm /etc/nginx/sites-enabled/default
```

Nakonec zbývá vytvořit a aktivovat službu *gunicorn*, která zprostředkovává komunikaci webového serveru s Python aplikací. Je potřeba vytvořit soubor `/etc/systemd/system/gunicorn.service` s obsahem

```
[Unit]
Description=gunicorn daemon
Requires=gunicorn.socket
After=network.target

[Service]
User=deploy
Group=deploy
WorkingDirectory=/srv/www/sos/sos
Environment="DJANGO_SETTINGS_MODULE=conf.settings.production"
ExecStart=/home/deploy/.pyenv/versions/sos/bin/gunicorn \
    --access-logfile - \
    --workers 2 \
```

```
    --bind unix:/run/gunicorn.sock \
    conf.wsgi:application

[Install]
WantedBy=multi-user.target
```

a soubor `/etc/systemd/system/gunicorn.socket` s obsahem

```
[Unit]
Description=gunicorn socket

[Socket]
ListenStream=/run/gunicorn.sock

[Install]
WantedBy=multi-user.target
```

a následně spustit následující příkazy:

```
systemctl enable gunicorn
systemctl start gunicorn
```

Posledním balíčkem, který je třeba nainstalovat, je `gettext`. Ten je potřebný pro správné fungování jazykové lokalizace systému SOS:

```
apt install gettext
```

A.3 Instalace a konfigurace SOS

Nejprve musí být vytvořen adresář `/srv/www/sos/` a jeho vlastnictví předáno uživateli *deploy*:

```
mkdir /srv/www/sos/
chown -R deploy:deploy /srv/www/sos/
```

Dalším krokem je ukončení připojení jako uživatel *root* a připojení se jako uživatel *deploy*. Přepínač `-A` v následujícím příkazu slouží pro aktivaci mechanizmu *agent forwarding* a v případě, že nebyl v prvním kroku tohoto manuálu vytvořen SSH klíč, není potřeba jej používat:

```
ssh <adresa_serveru> -l deploy -A
```

Ještě před samotnou instalací systému SOS je potřeba zprovoznit virtuální prostředí pro jazyk Python. Prvním krokem je instalace aplikace Pyenv a zásuvného modulu Pyenv-virtualenv podle návodů dostupných na adresách `https://github.com/pyenv/pyenv`, resp. `https://github.com/pyenv/pyenv-virtualenv`. Aplikaci je nutné nainstalovat do adresáře `/.pyenv/`.

Dalším krokem je instalace jazyka Python verze 3.9.4 a vytvoření virtuálního prostředí pro systém SOS:

```
pyenv install 3.9.4
pyenv virtualenv 3.9.4 sos
pyenv activate sos
```

Nyní je třeba na server do složky `/srv/www/sos/` přenést zdrojové kódy systému SOS. Lze tak učinit buďto naklonováním GitLab repozitáře, nebo zkopírováním souborů z přiloženého paměťového média. K naklonování repozitáře lze použít následující příkaz:

```
git clone git@gitlab.fit.cvut.cz:pavlutom/sos.git && cd sos
```

Dále je potřeba nainstalovat do virtuálního prostředí všechny potřebné Python balíčky následujícím příkazem:

```
pip install -r requirements/production.txt
```

Předposledním krokem je vytvoření souboru `.env` v adresáři projektu a definovat zde hodnoty proměnných prostředí. Soubor musí obsahovat následující definice:

```
DEBUG=False
STAGE=False                    # pro testovací prostředí 'True'
SECRET_KEY=<náhodný_retezec>

PROJECT_NAME=SOS
PROJECT_URL=<adresa_serveru>

OAUTH_CLIENT_ID=<hodnota>      # z administrátorského rozhraní
OAUTH_CLIENT_SECRET=<hodnota> # fakultního OAuth 2.0 serveru
JWT_SECRET_KEY=<náhodný_retezec>

DB_NAME=sos
DB_USER=sos
DB_PASS=<heslo_od_databáze>
DB_HOST=localhost
DB_PORT=5432
DB_ATOMIC_REQUESTS=True
```

Posledním krokem instalace je spuštění skriptu, který obsahuje příkazy nutné k uvedení systému do provozu. Tento skript je také spouštěn vzdáleně v rámci CI/CD.

```
./deploy_actions.sh
```

Pro použití systému ještě vytvořit administrátorský účet spuštěním následujícího příkazu:

```
./manage.py createsuperuser
```

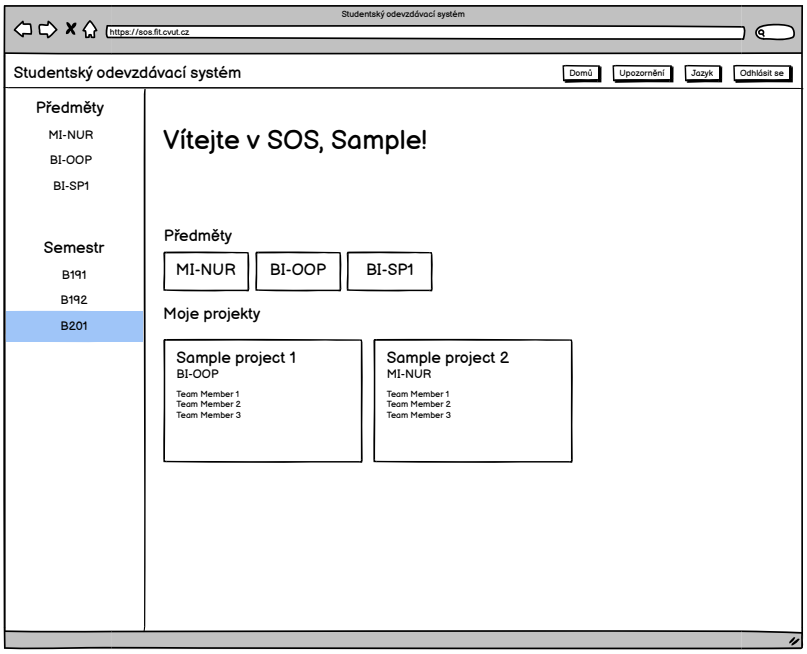
Po spuštění příkazu bude uživatel vyzván k zadání přihlašovacích údajů pro tento účet.

Nyní je systém SOS připraven k použití.

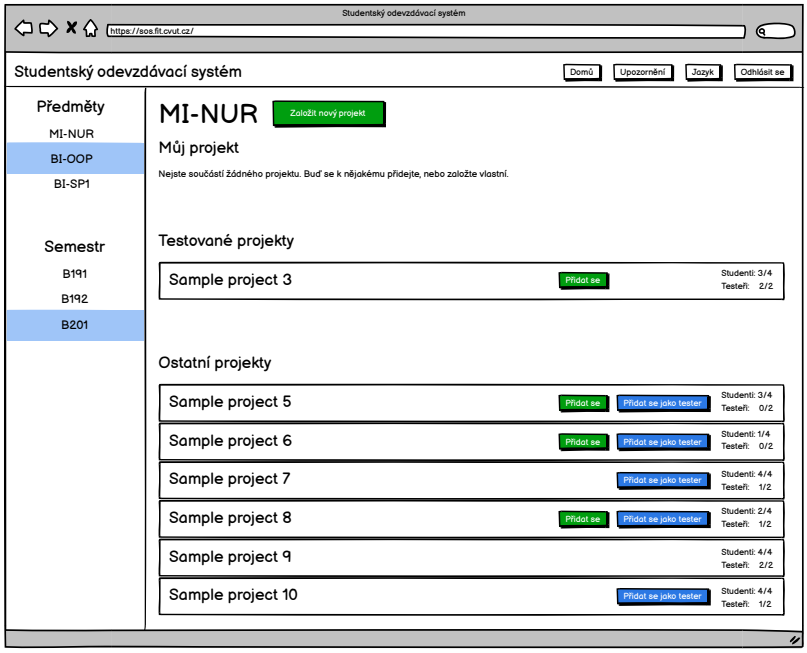
Příloha B

Drátěné modely obrazovek navržené aplikace

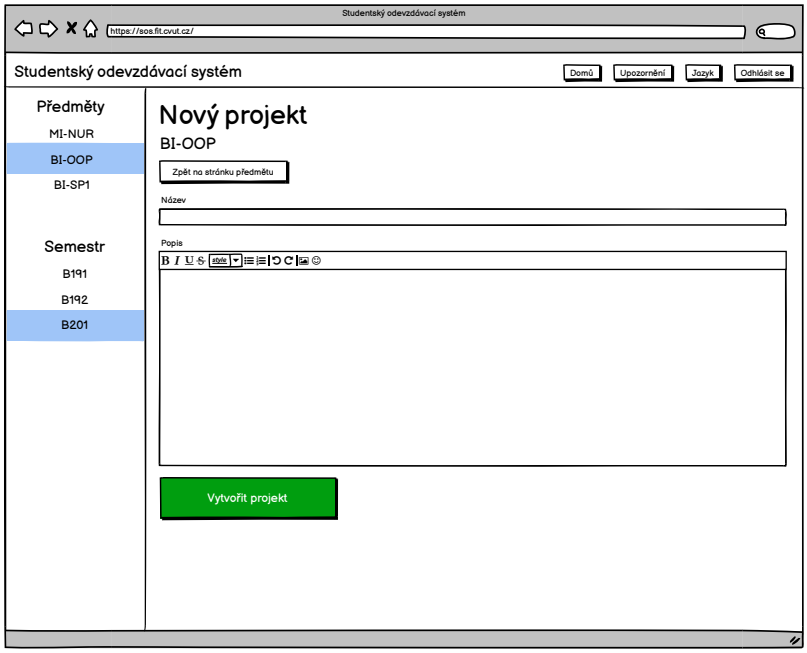
Tato příloha obsahuje drátěné modely jednotlivých obrazovek aplikace, na které se odkazuje sekce 2.7.2.



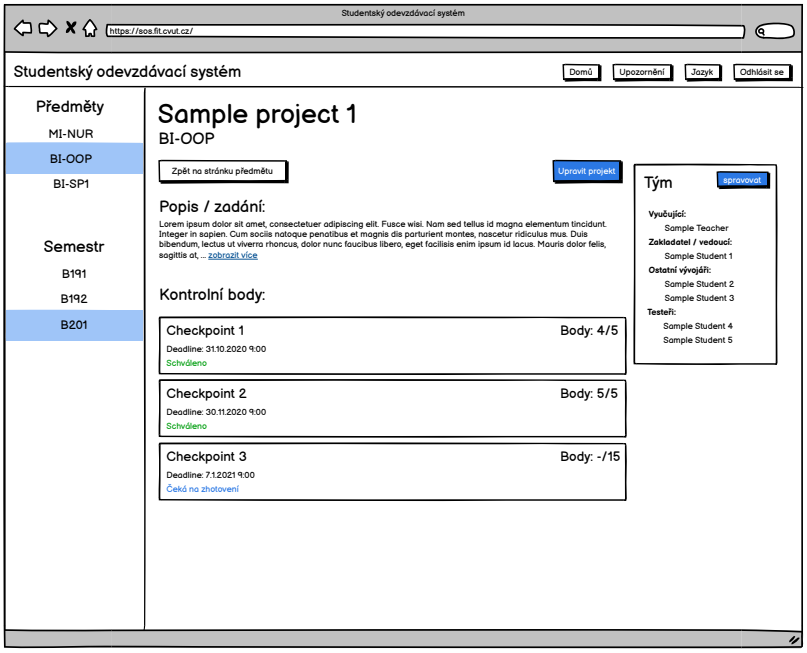
Obrázek B.1: Domovská stránka studenta – wireframe



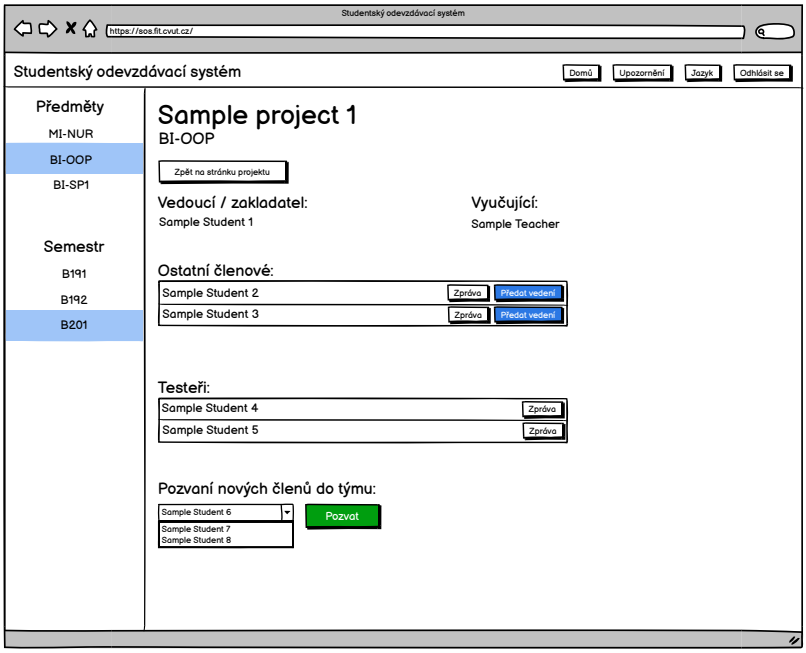
Obrázek B.2: Stránka předmětu z pohledu studenta – wireframe



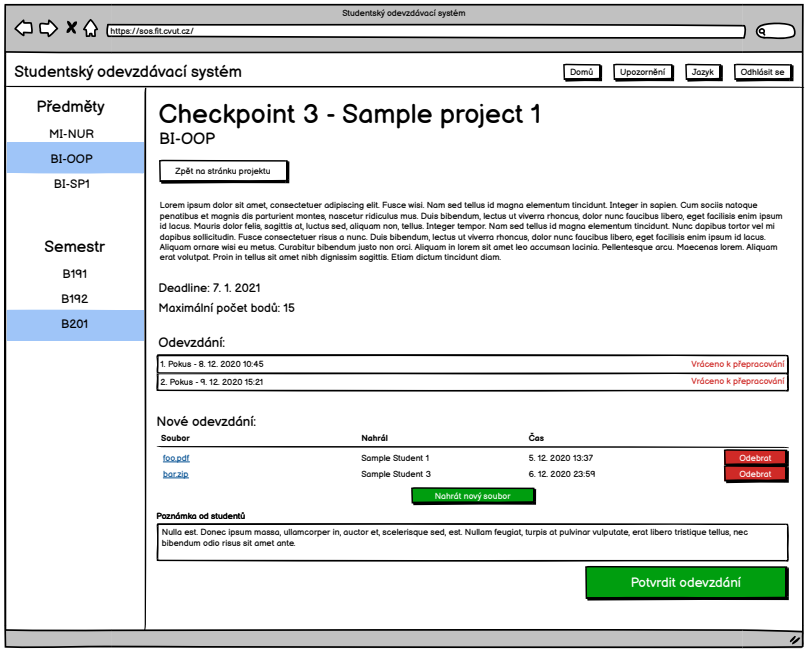
Obrázek B.3: Stránka tvorby/úpravy zadání projektu studentem – wireframe



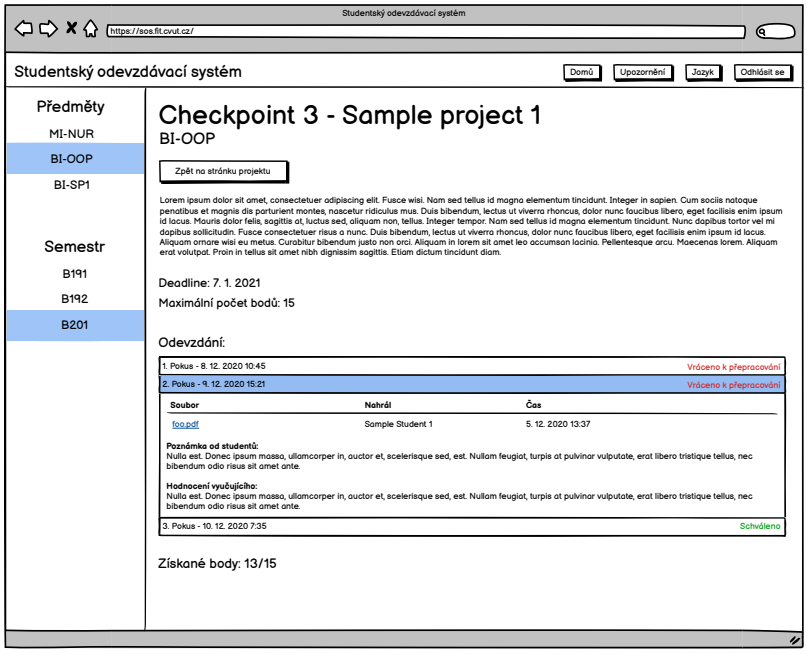
Obrázek B.4: Stránka projektu z pohledu studenta – wireframe



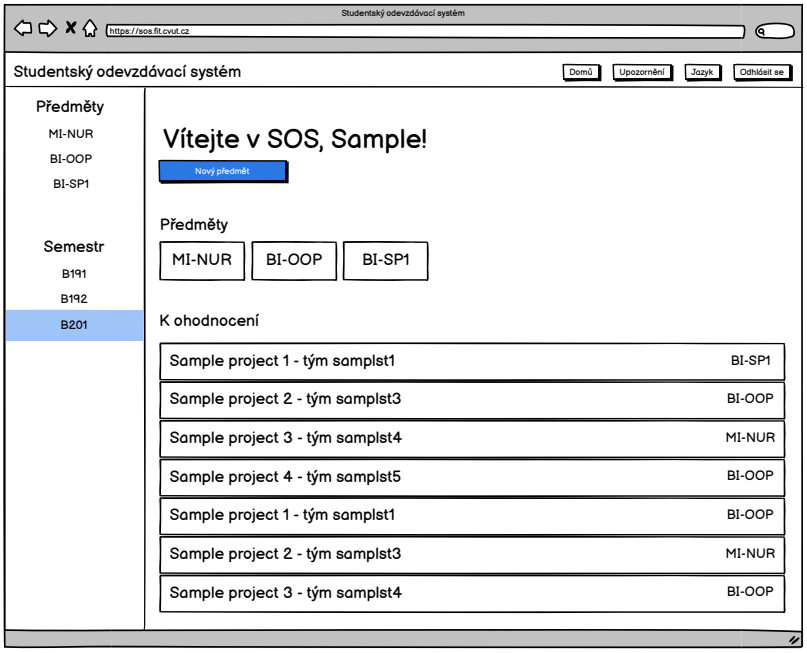
Obrázek B.5: Stránka správy týmu z pohledu studenta – wireframe



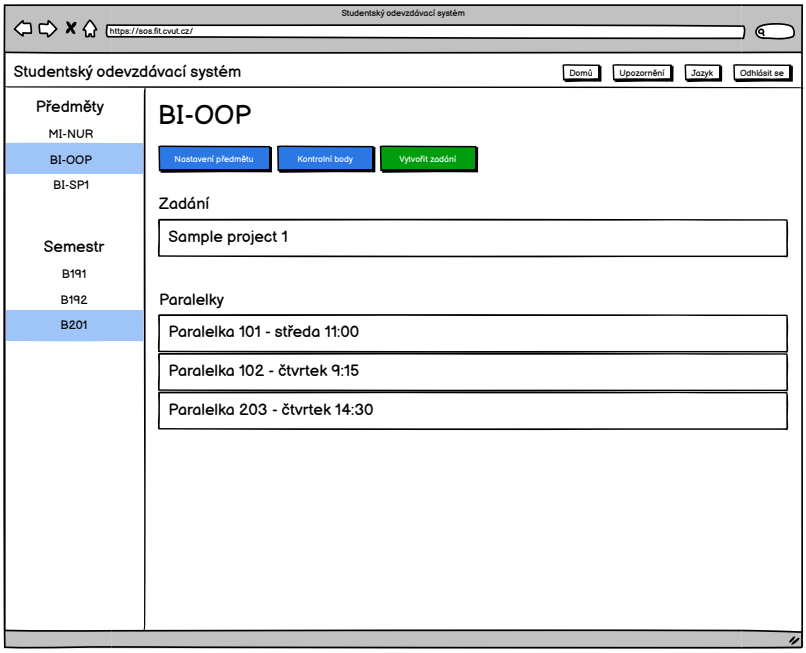
Obrázek B.6: Stránka kontrolního bodu pohledu studenta – wireframe



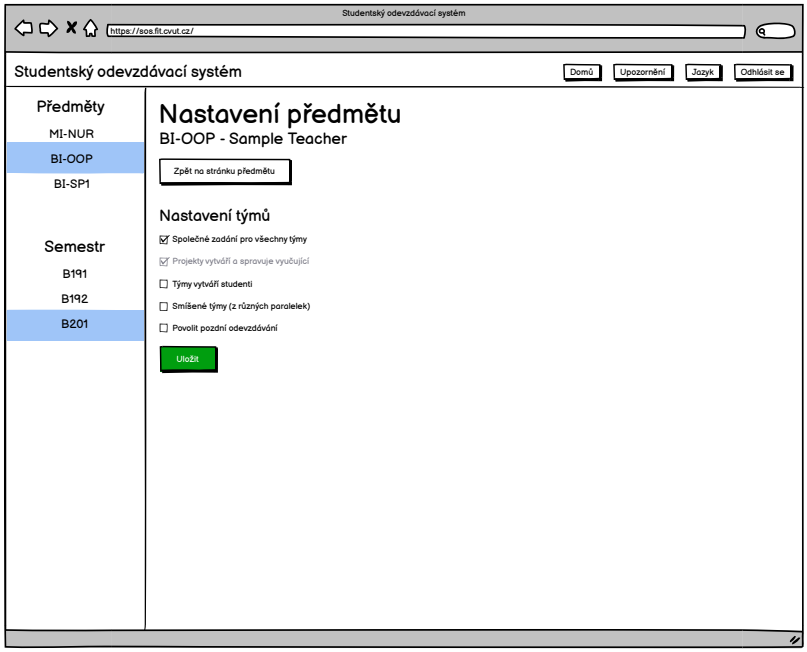
Obrázek B.7: Stránka kontrolního bodu s hodnocením z pohledu studenta – wireframe



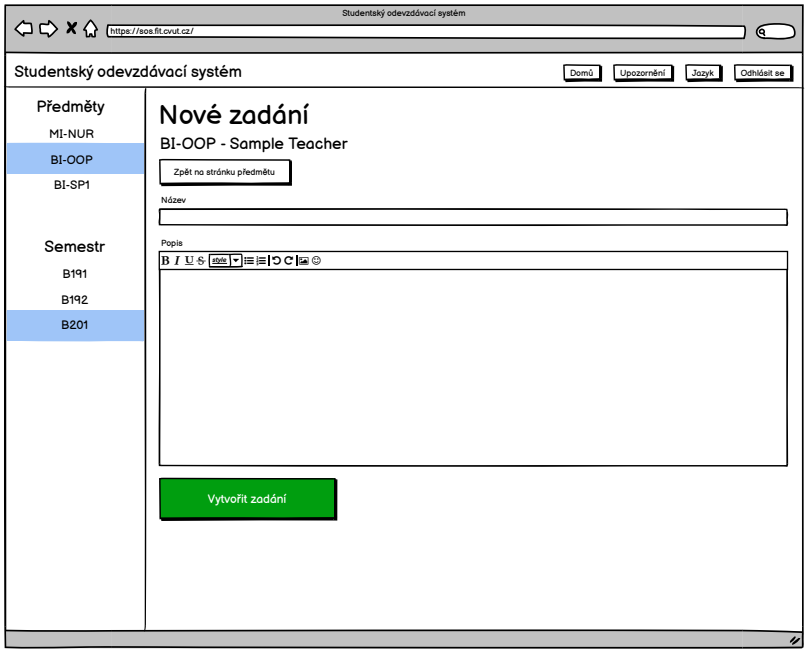
Obrázek B.8: Domovská stránka vyučujícího – wireframe



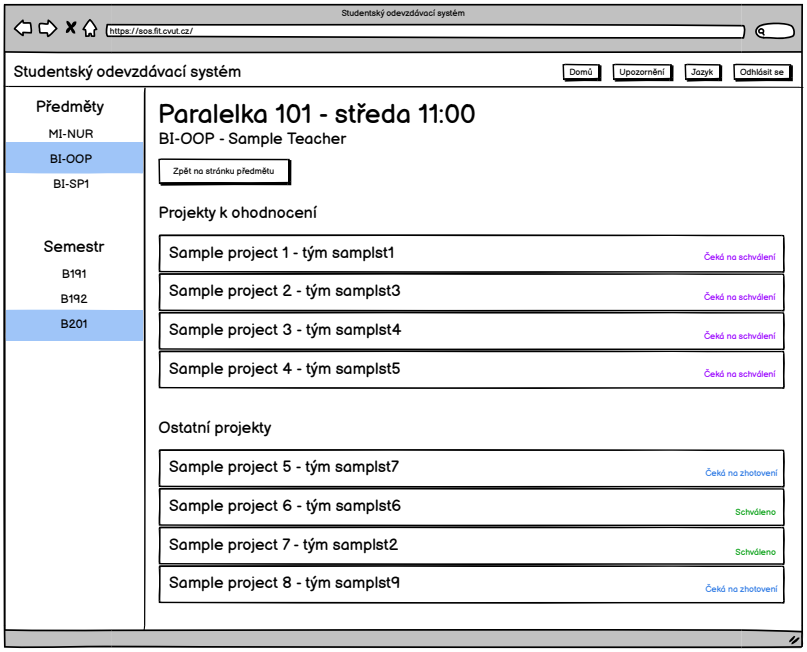
Obrázek B.9: Stránka předmětu z pohledu vyučujícího – wireframe



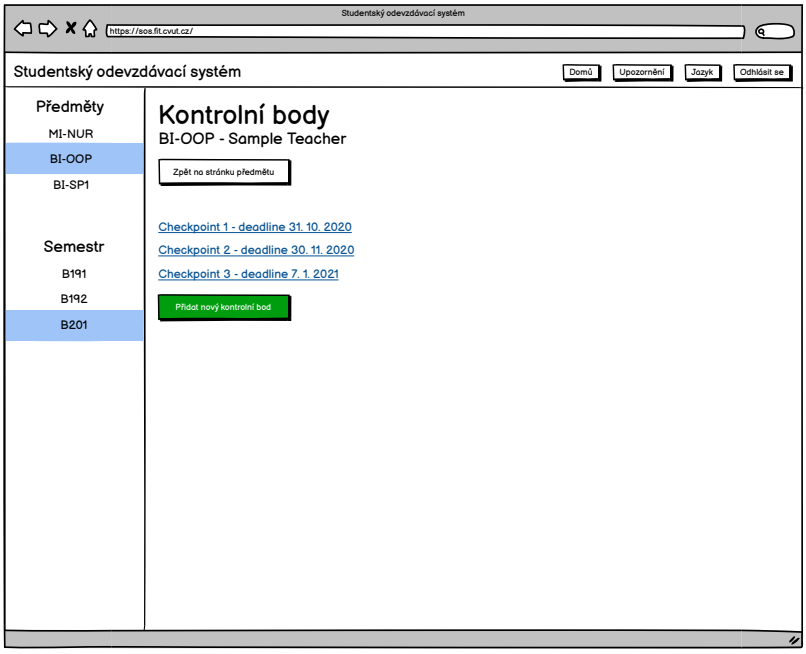
Obrázek B.10: Stránka konfigurace předmětu z pohledu vyučujícího – wireframe



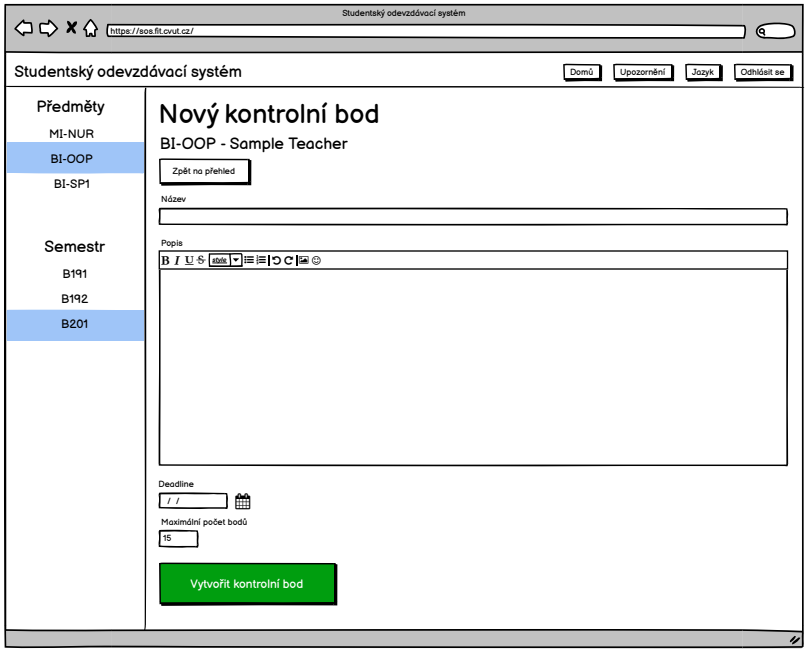
Obrázek B.11: Stránka tvorby/úpravy zadání vyučujícím – wireframe



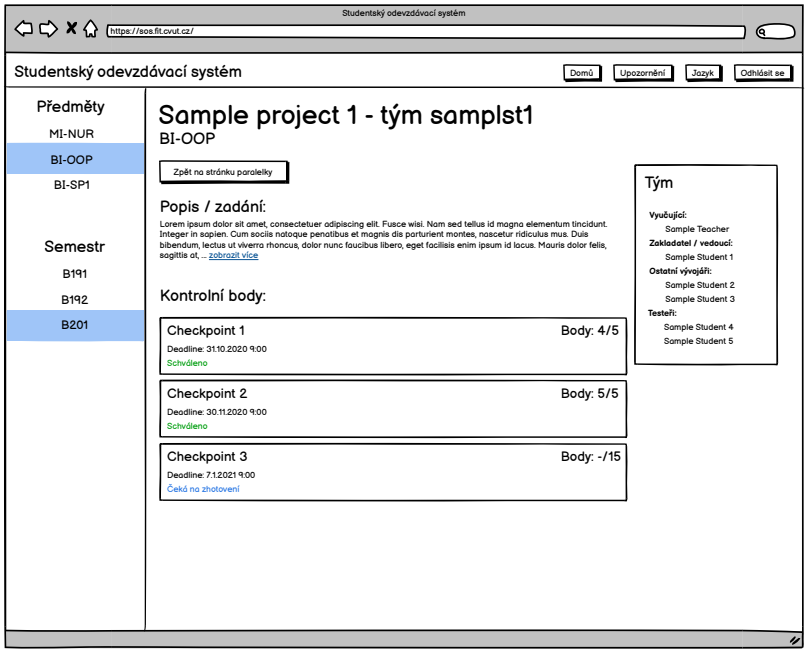
Obrázek B.12: Stránka paralelky z pohledu vyučujícího – wireframe



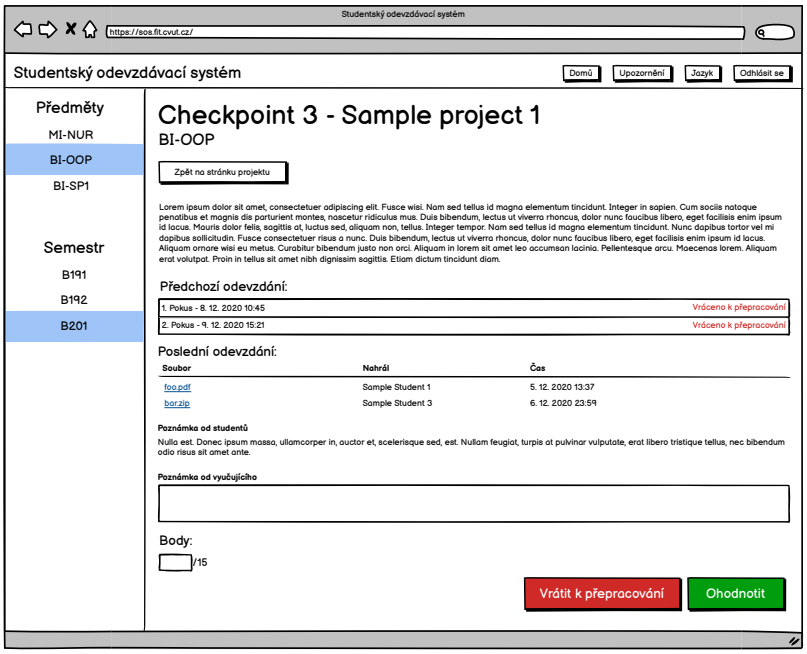
Obrázek B.13: Stránka s přehledem kontrolních bodů z pohledu vyučujícího – wireframe



Obrázek B.14: Stránka tvorby/úpravy kontrolního bodu vyučujícím – wireframe



Obrázek B.15: Stránka projektu z pohledu vyučujícího – wireframe



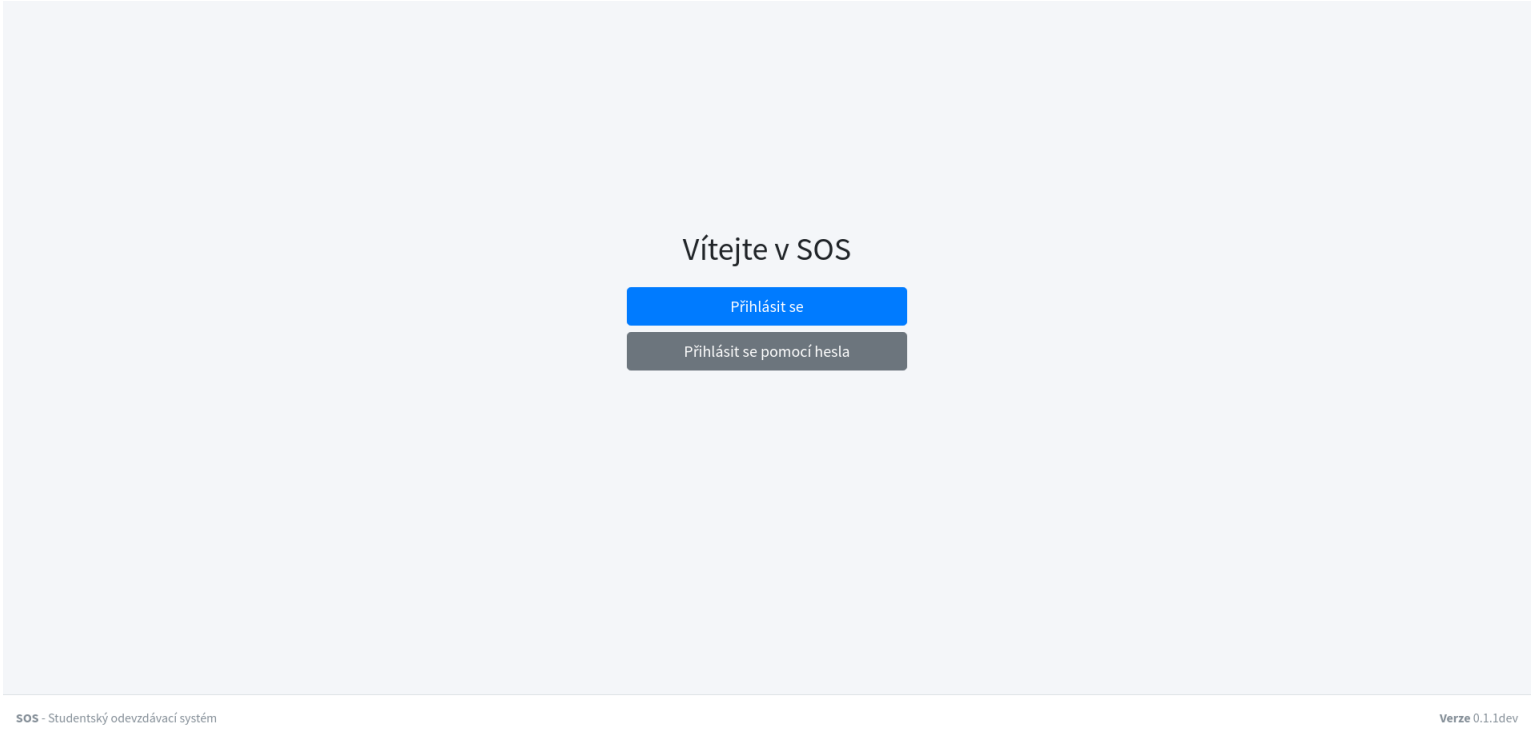
Obrázek B.16: Stránka kontrolního bodu v rámci projektu z pohledu vyučujícího – wireframe

Příloha C

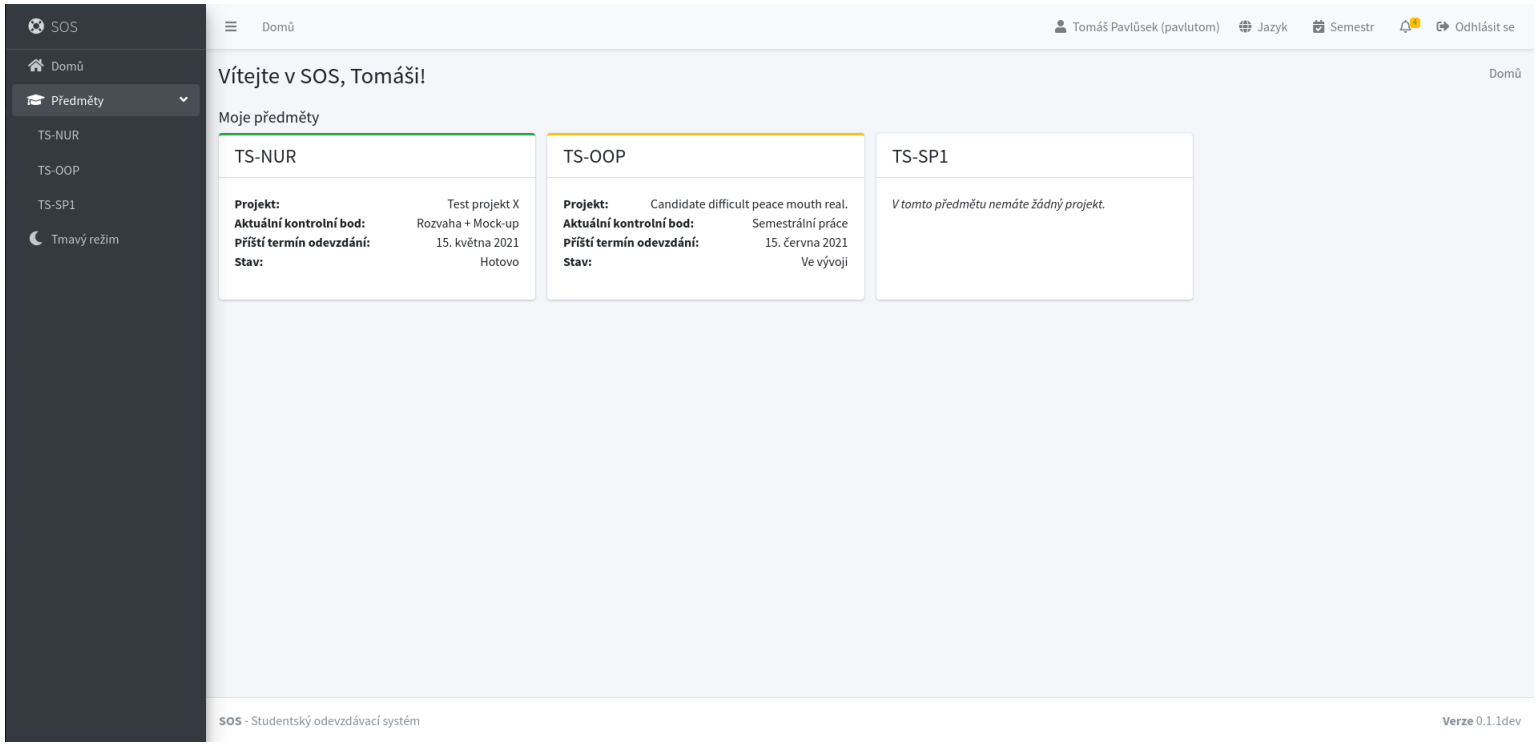
Obrazovky realizované aplikace

Tato příloha obsahuje snímky jednotlivých obrazovek aplikace, na které se odkazuje sekce 3.6.4. Tyto snímky dokumentují uživatelské rozhraní systému SOS ve stavu, v jakém bylo před zahájením uživatelského testování. Příloha obsahuje snímky obrazovek společné části uživatelského rozhraní, uživatelského rozhraní studenta a uživatelského rozhraní vyučujícího.

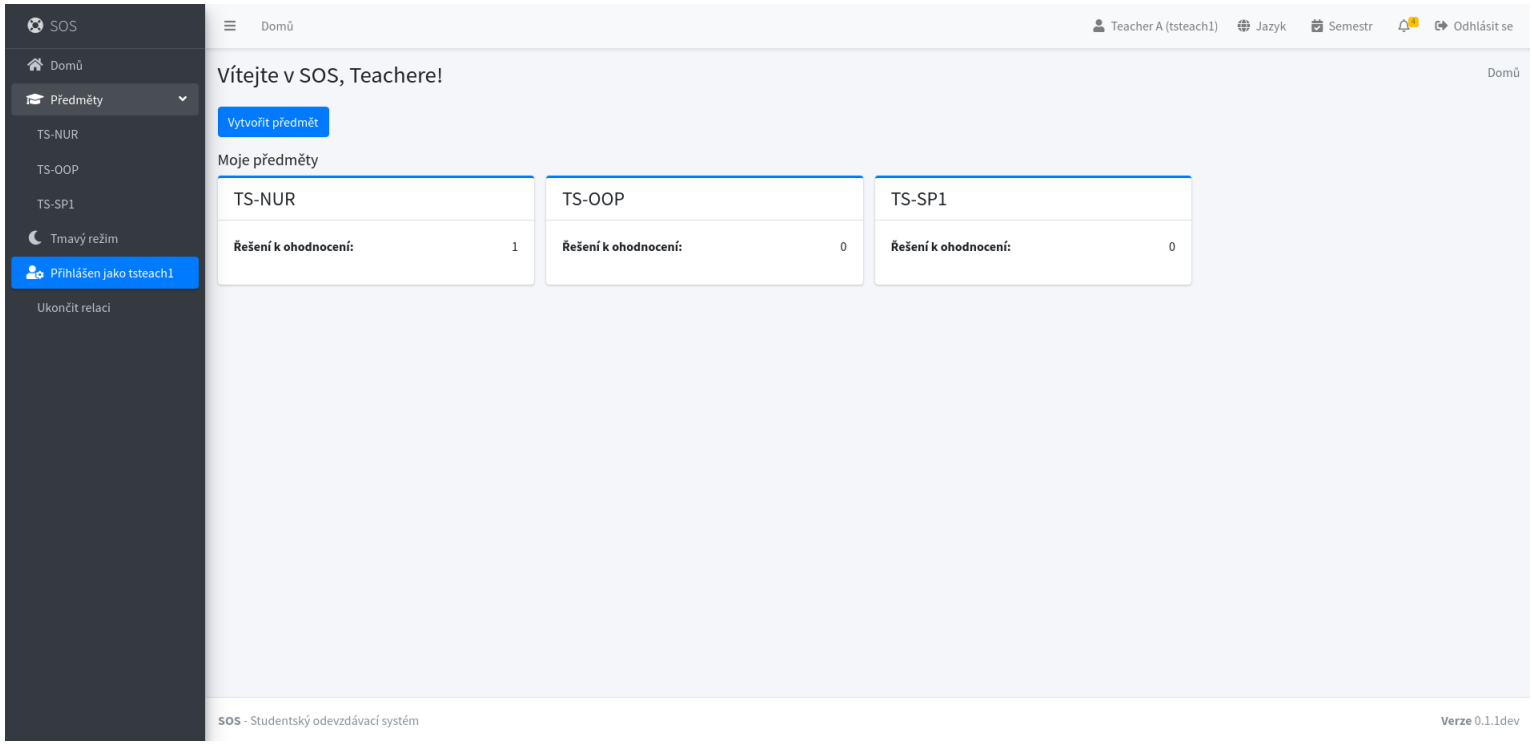
Snímky obrazovky z pohledu vyučujícího byly vytvořeny z pohledu administrátora impersonujicího vyučujícího, pročež se v postranním panelu navíc vyskytují položky „Přihlášen jako tsteach1“ a „Ukončit relaci“.



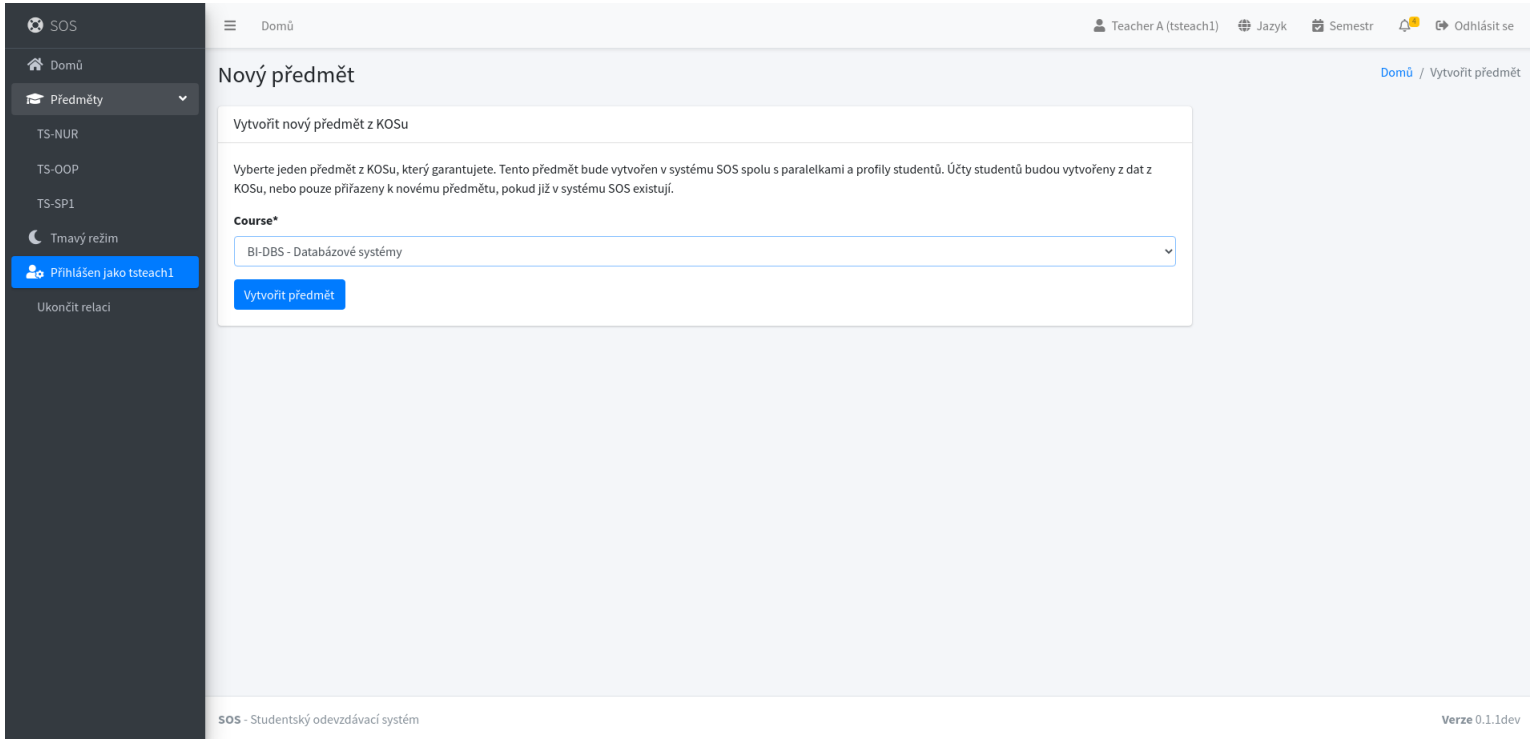
Obrázek C.1: Přihlašovací obrazovka – snímek obrazovky



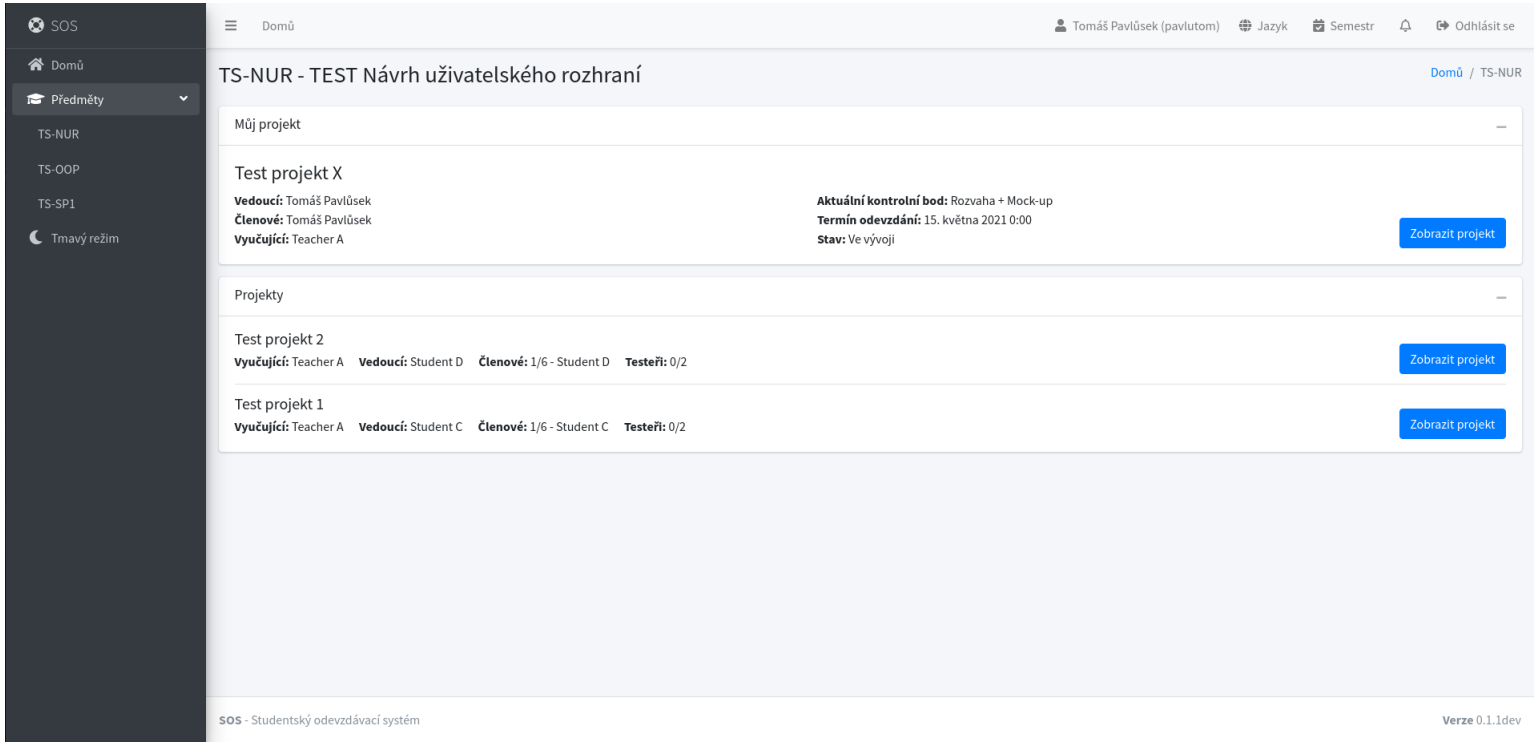
Obrázek C.2: Domovská stránka studenta – snímek obrazovky



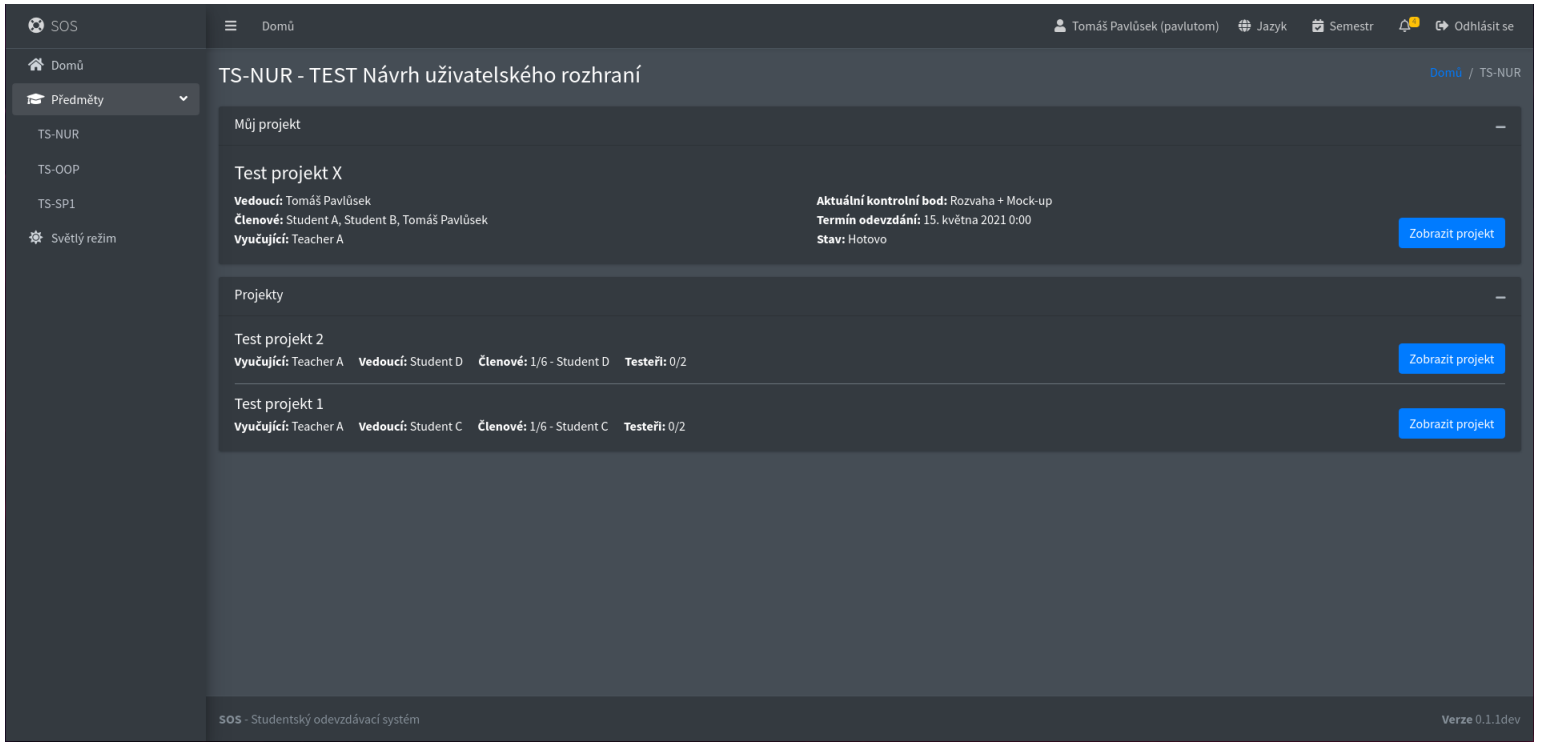
Obrázek C.3: Domovská stránka vyučujícího – snímek obrazovky



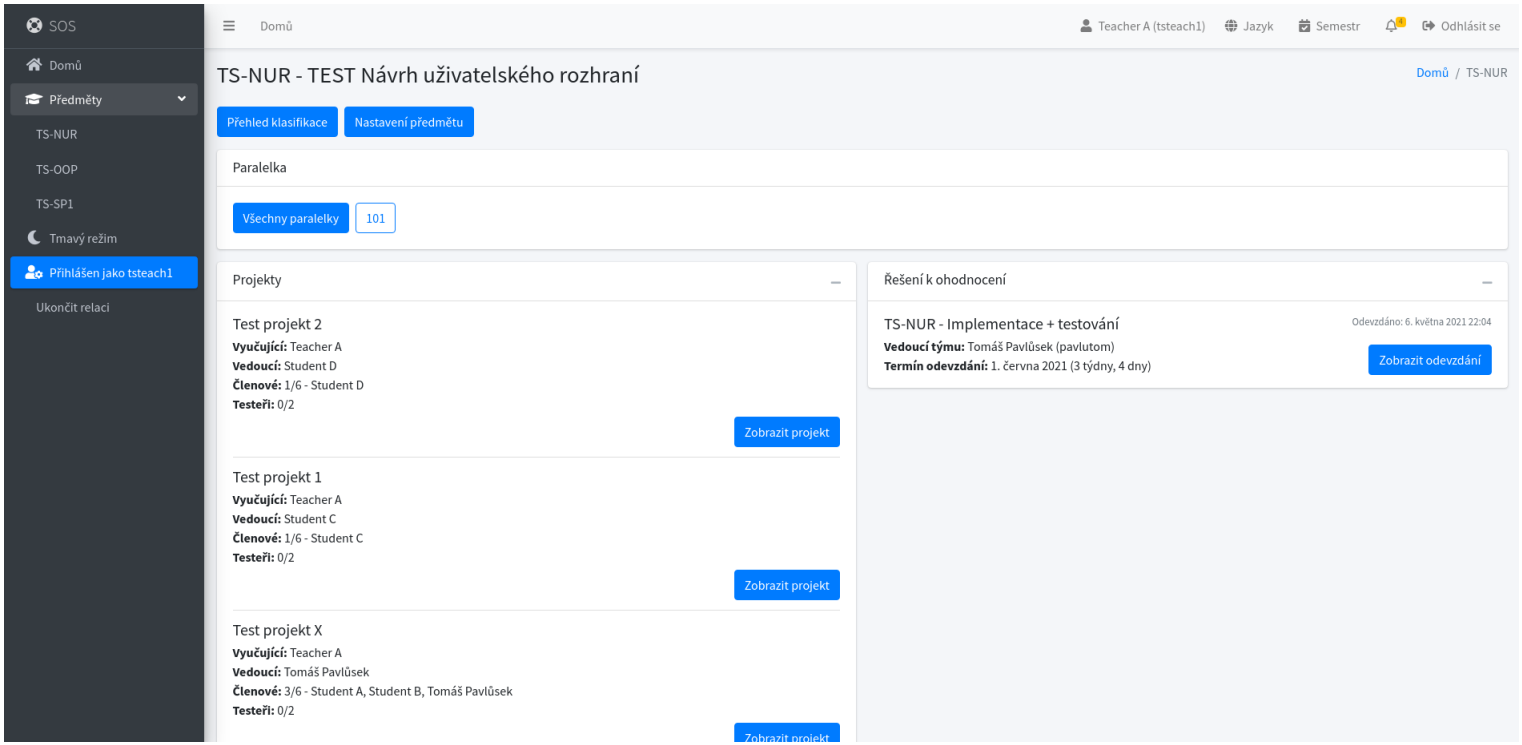
Obrázek C.4: Stránka tvorby předmětu – snímek obrazovky



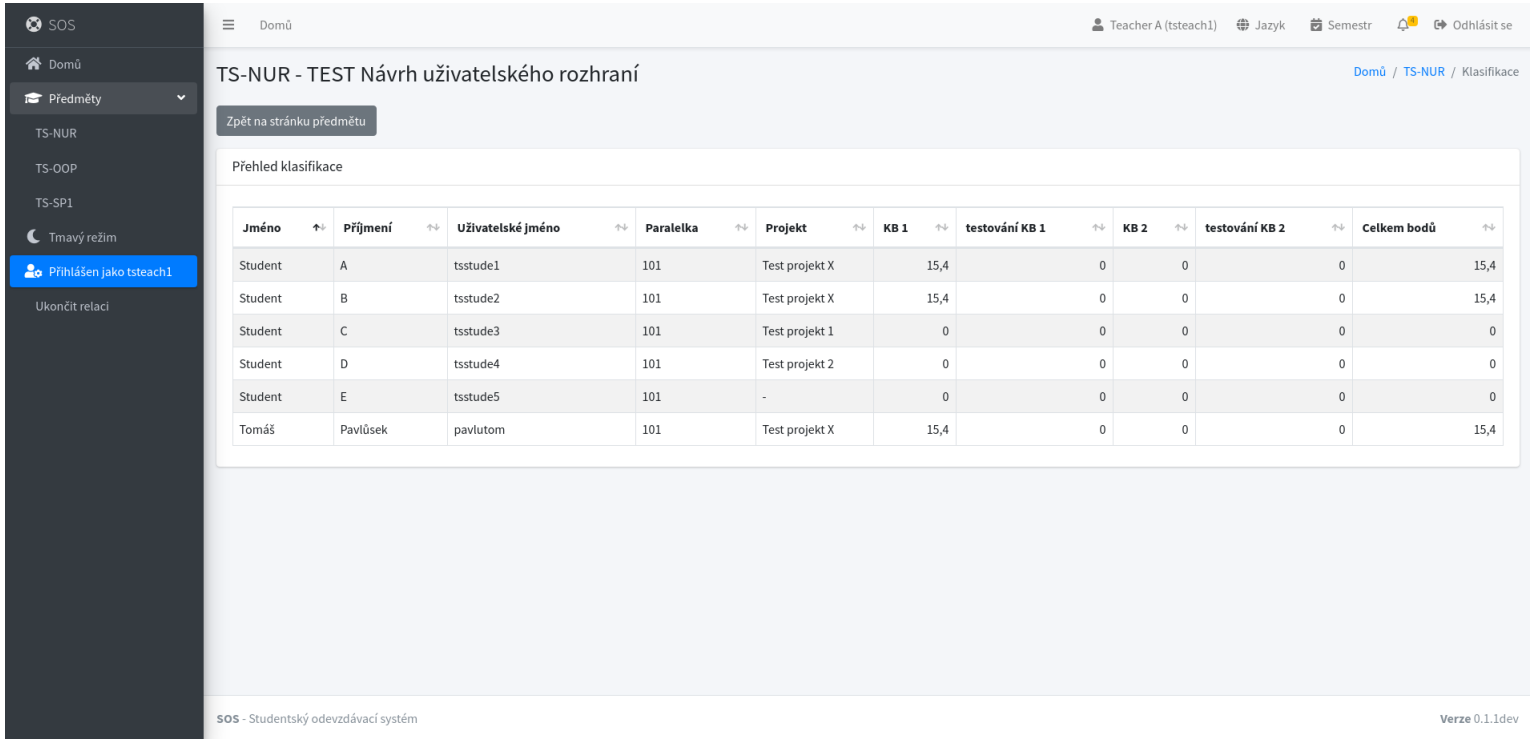
Obrázek C.5: Stránka předmětu z pohledu studenta – snímek obrazovky



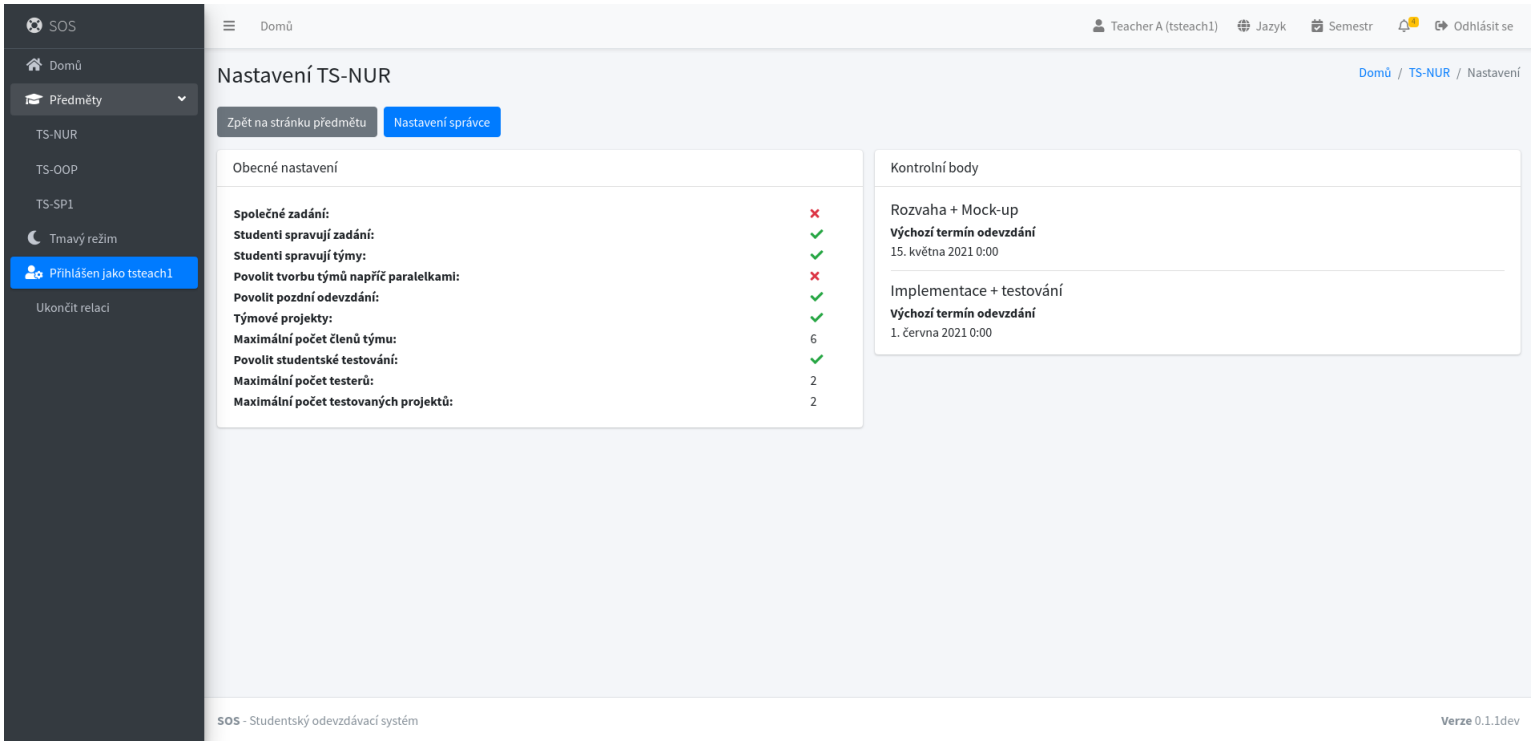
Obrázek C.6: Stránka předmětu z pohledu studenta (tmavý režim) – snímek obrazovky



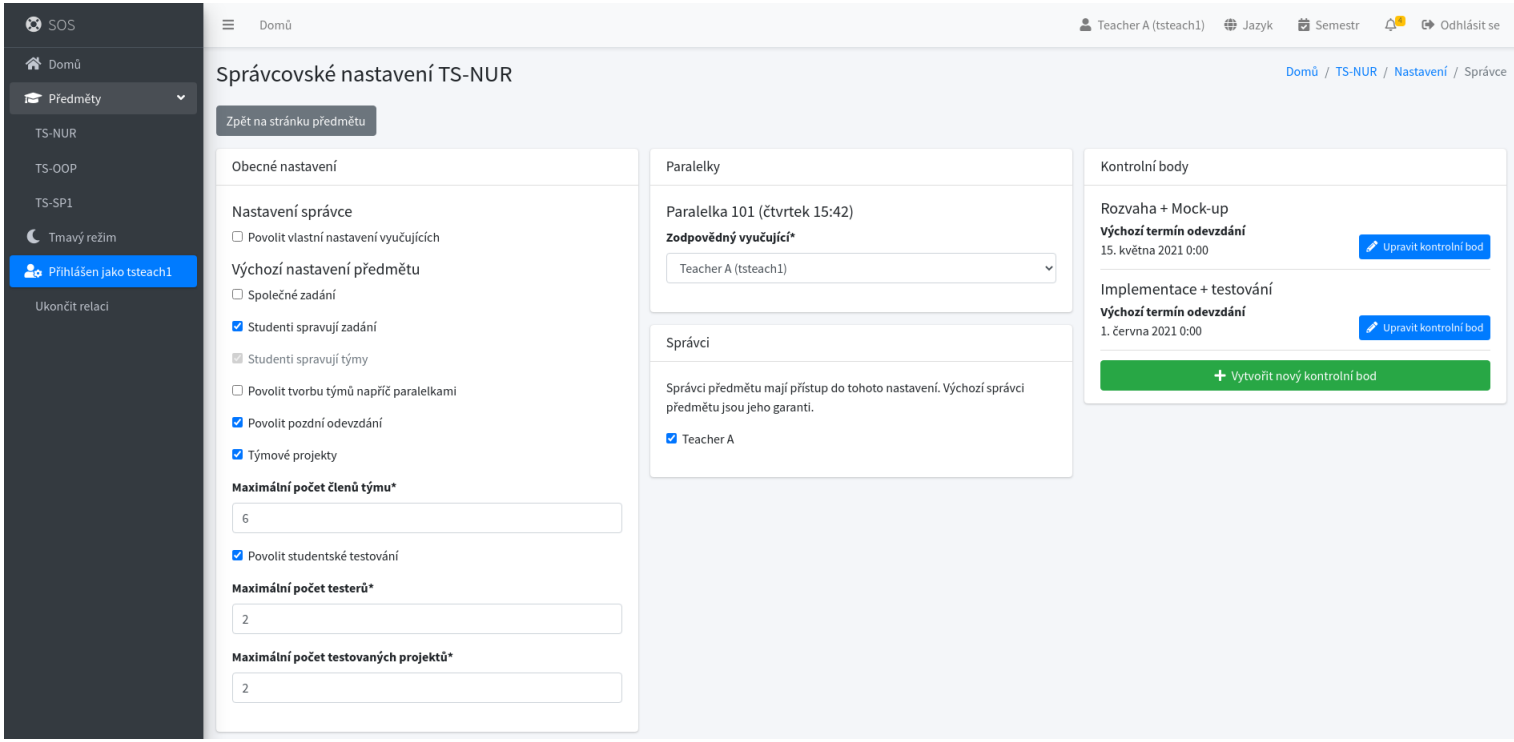
Obrázek C.7: Stránka předmětu z pohledu vyučujícího – snímek obrazovky



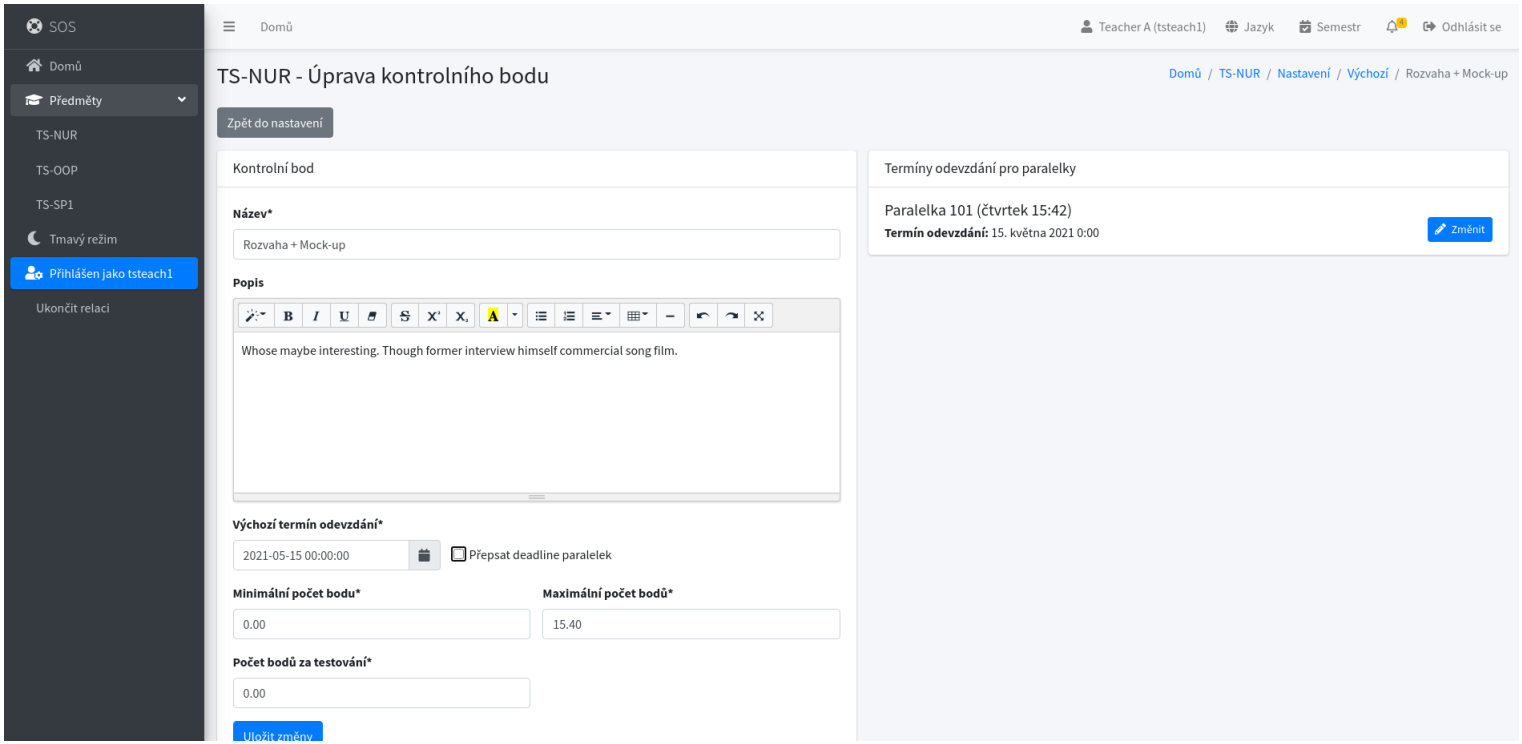
Obrázek C.8: Stránka přehledu klasifikace – snímek obrazovky



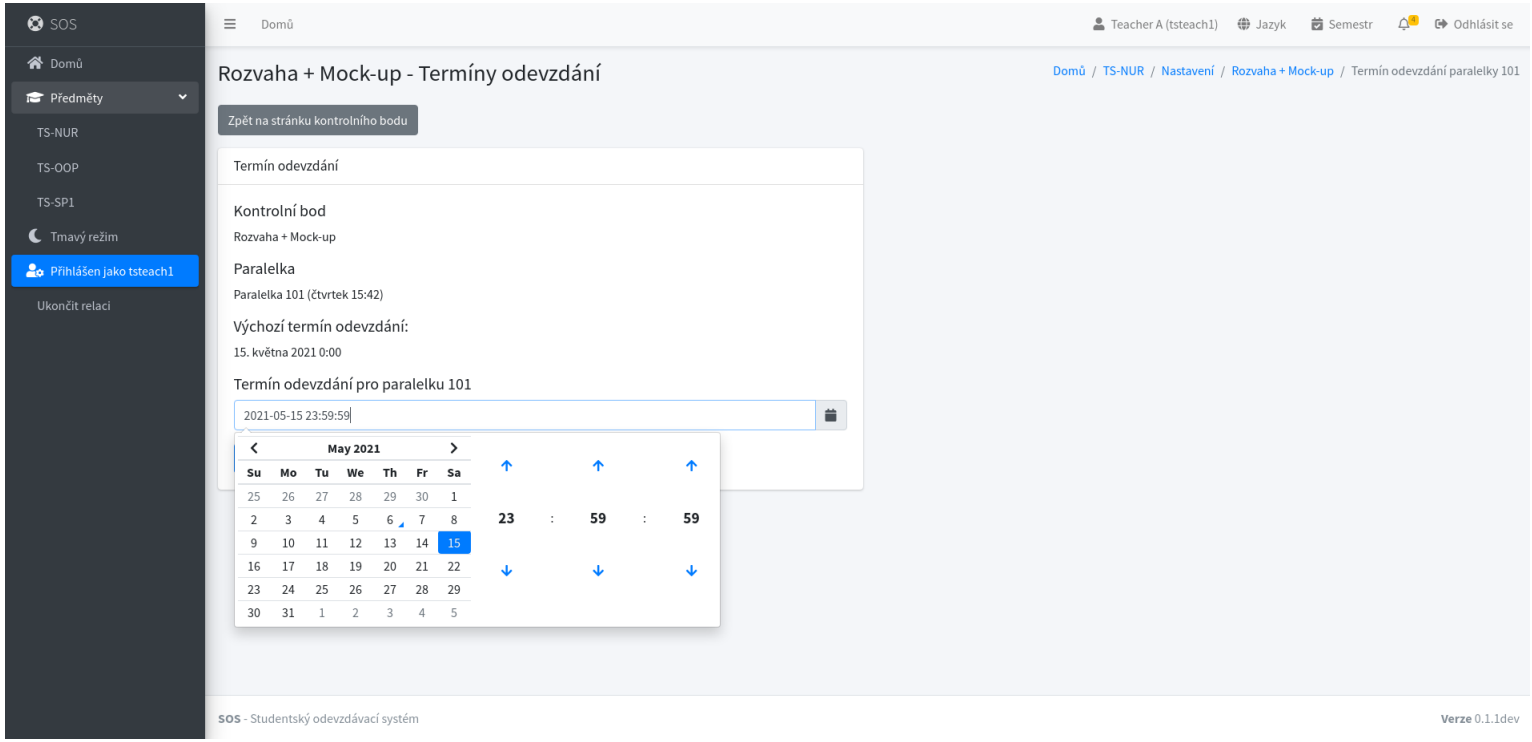
Obrázek C.9: Stránka nastavení předmětu (jen pro čtení) – snímek obrazovky



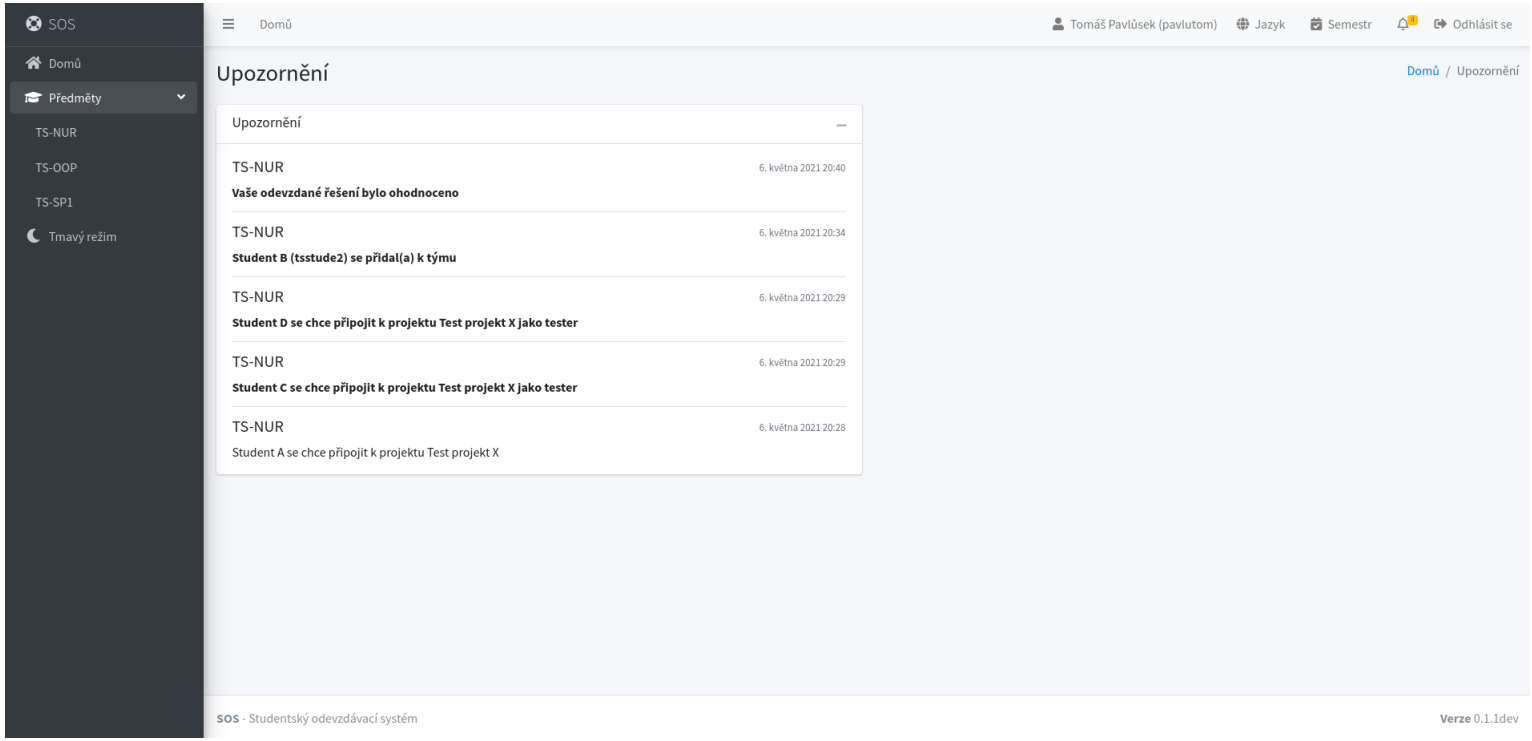
Obrázek C.10: Stránka nastavení předmětu (správce) – snímek obrazovky



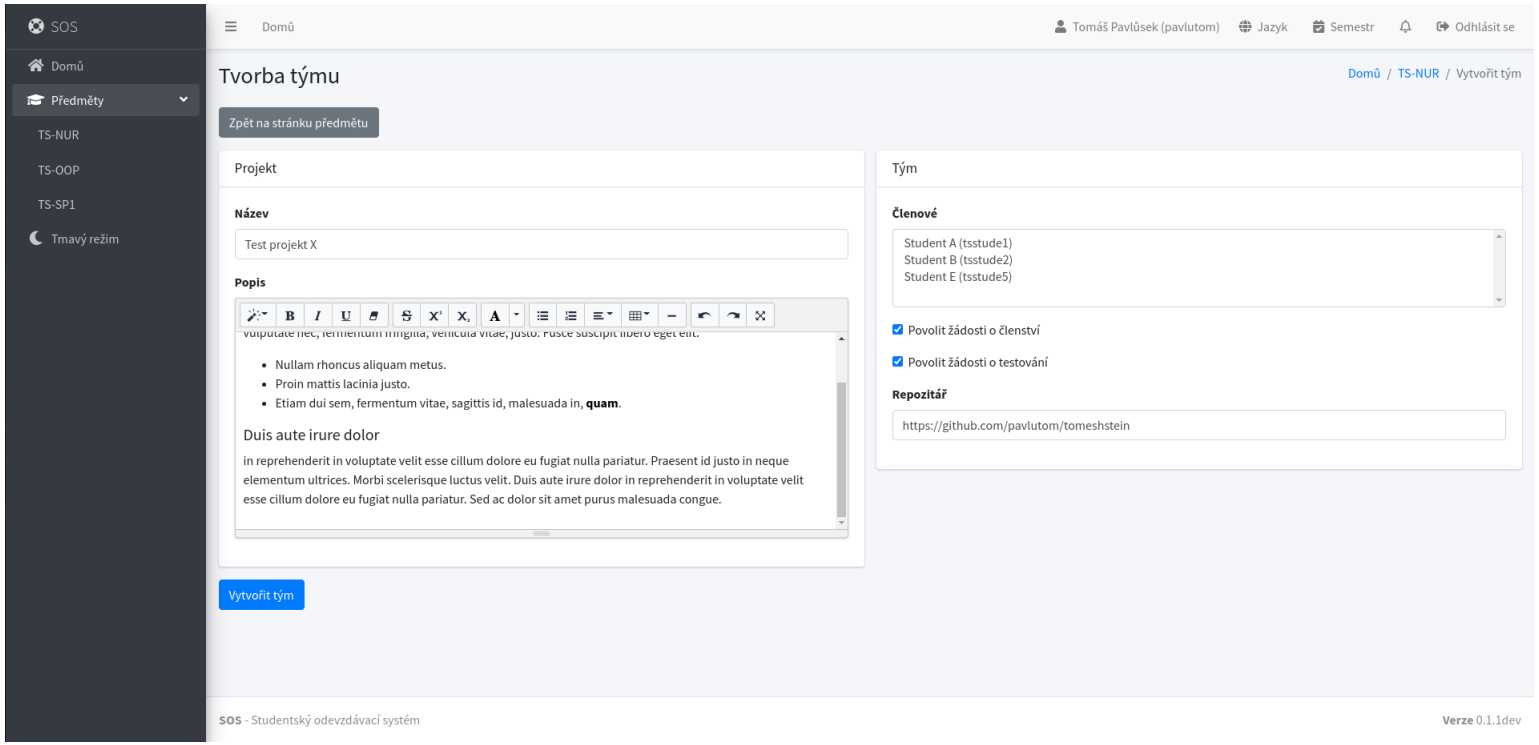
Obrázek C.11: Stránka úpravy kontrolního bodu – snímek obrazovky



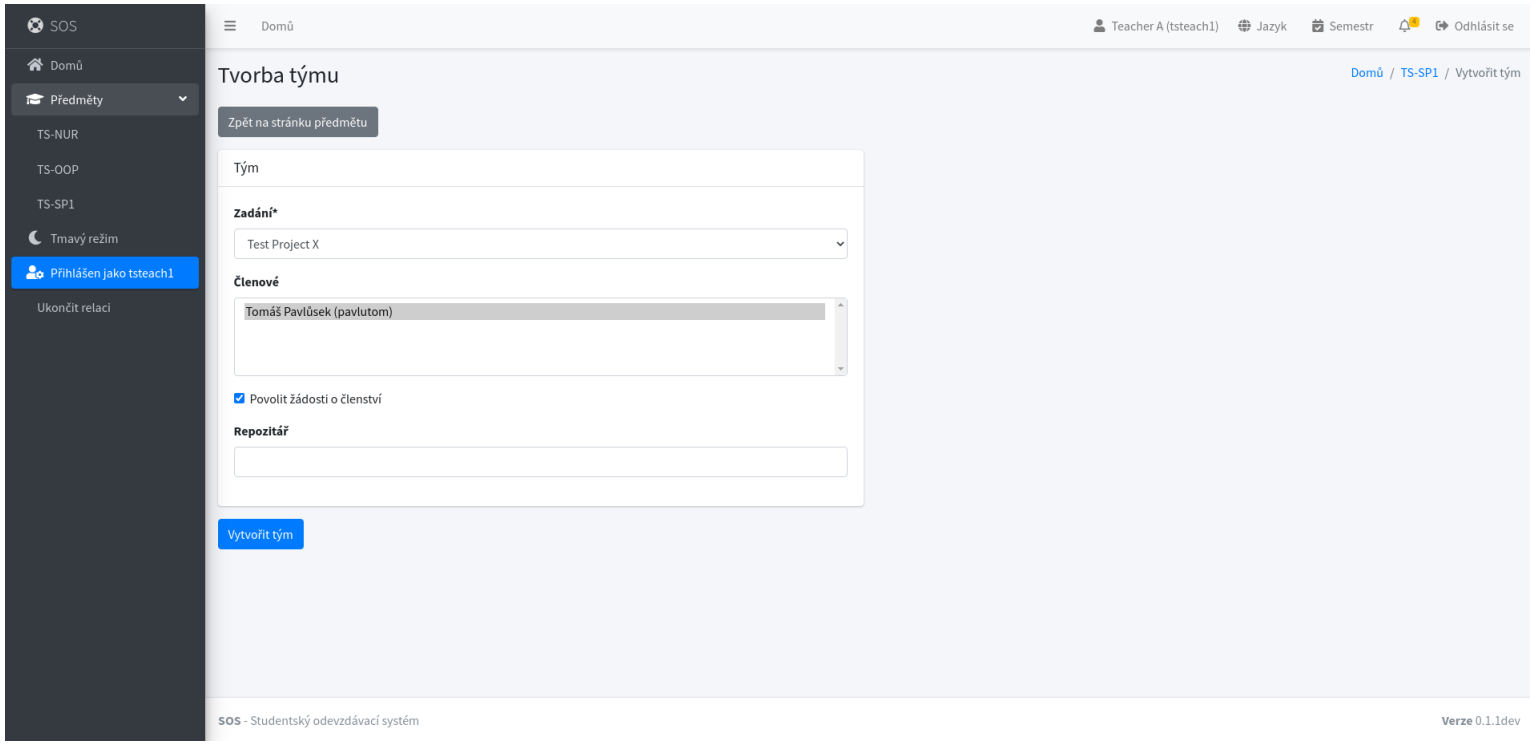
Obrázek C.12: Stránka úpravy termínu odevzdání paralelky – snímek obrazovky



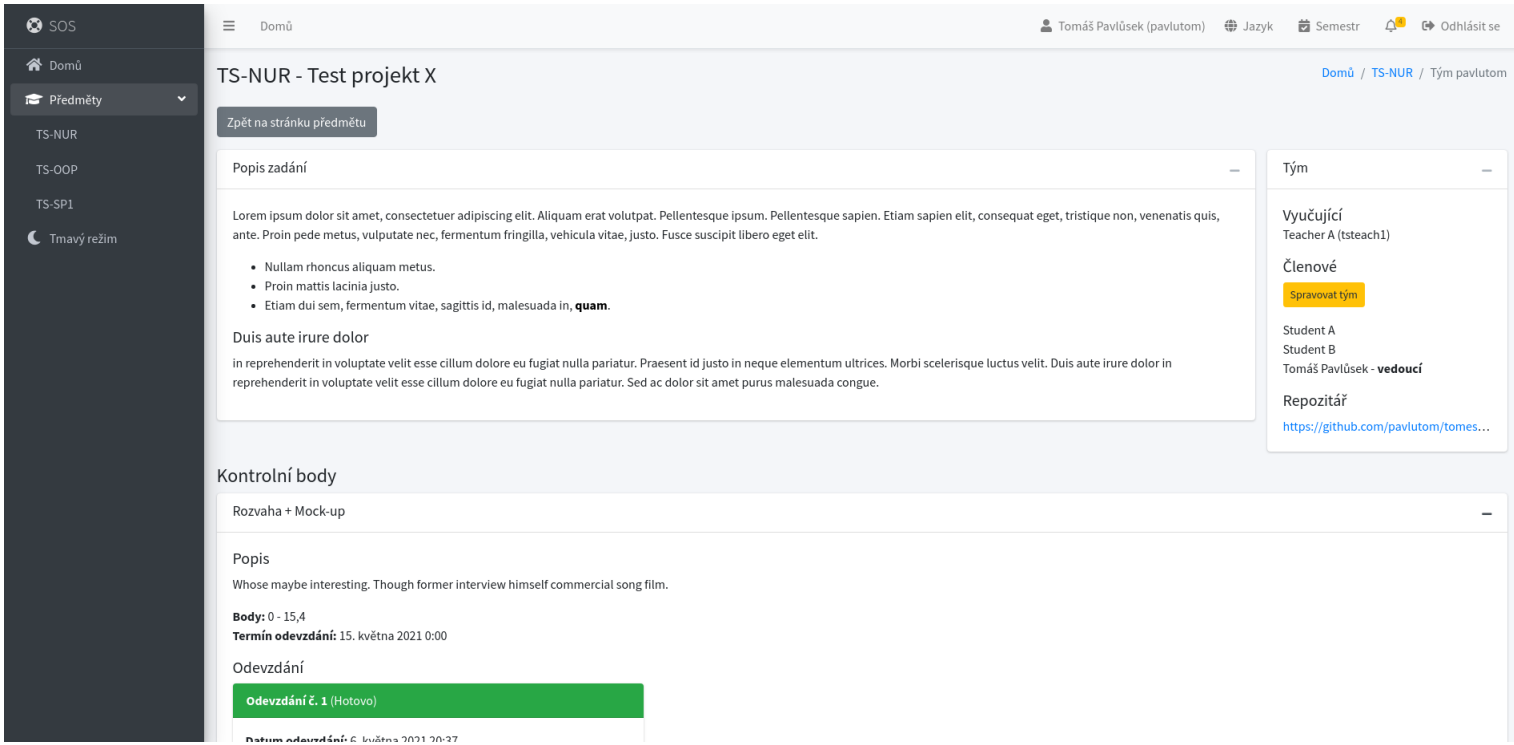
Obrázek C.13: Seznam notifikací – snímek obrazovky



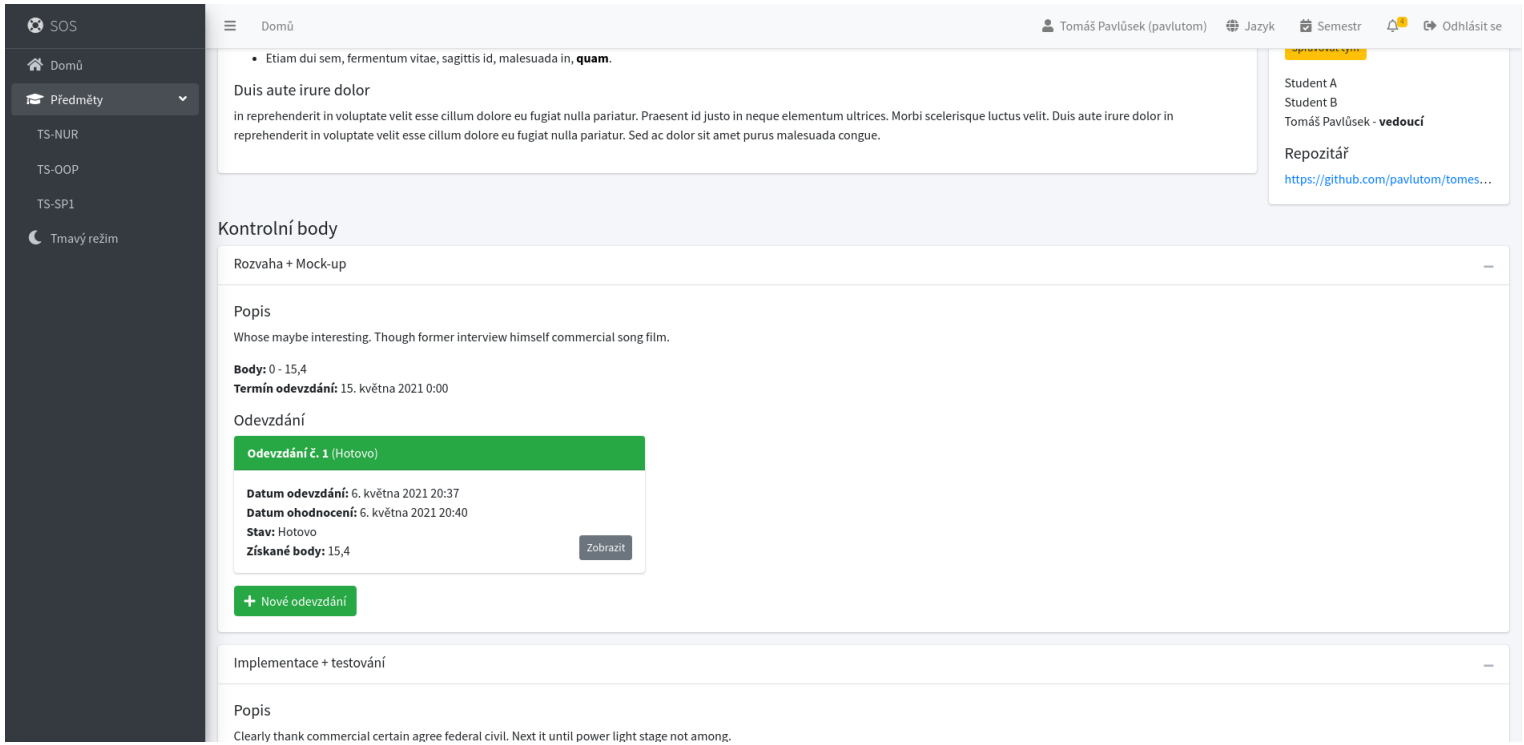
Obrázek C.14: Stránka tvorby projektu/týmu se zadáním – snímek obrazovky



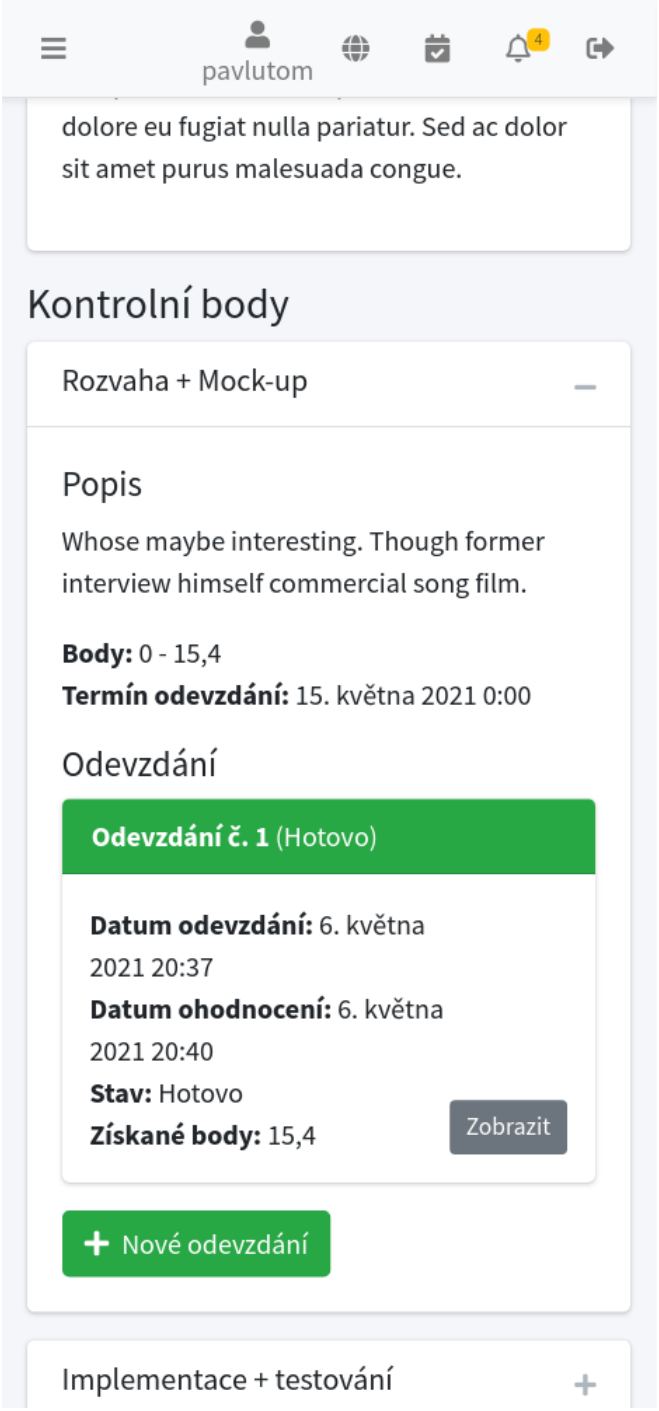
Obrázek C.15: Stránka tvorby projektu/týmu bez zadání – snímek obrazovky



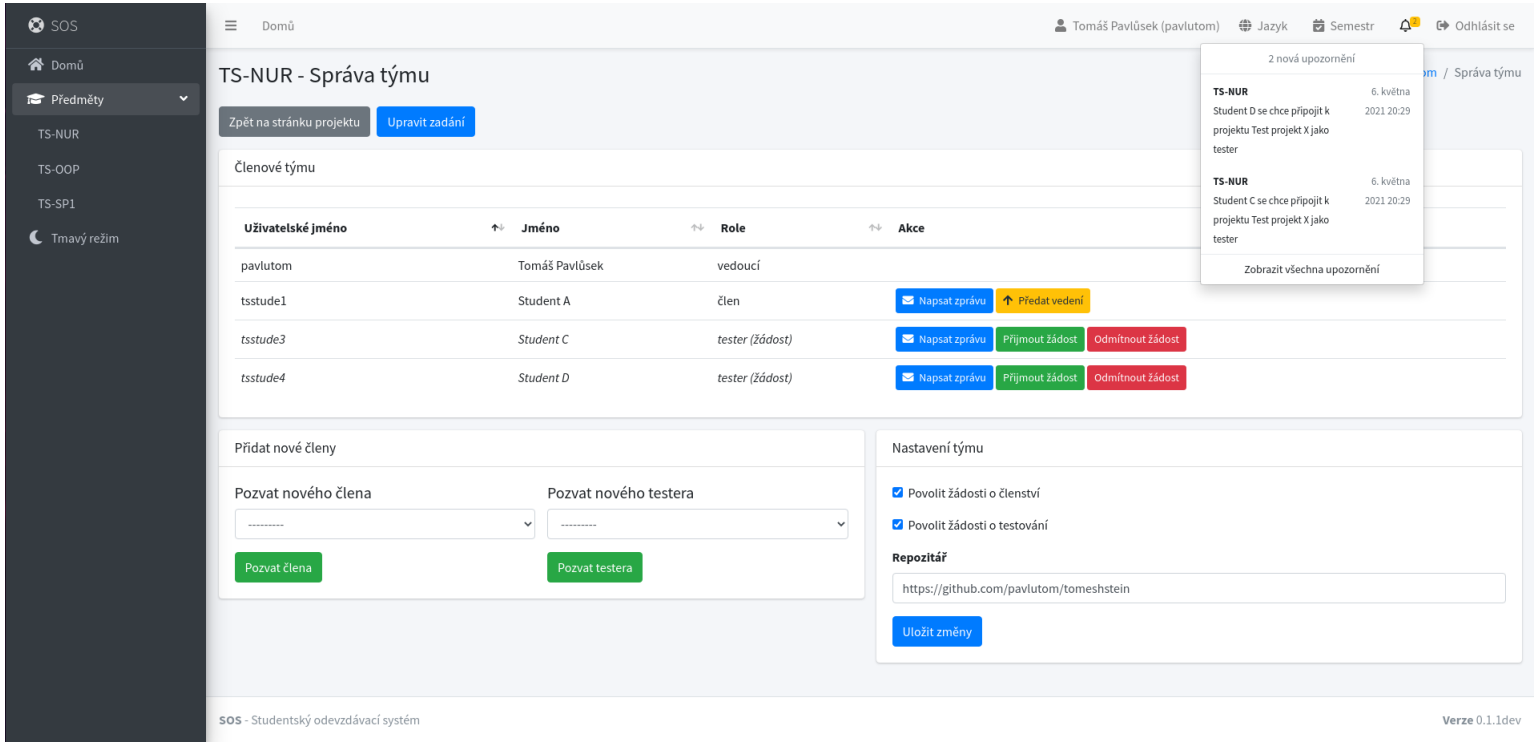
Obrázek C.16: Stránka projektu – snímek obrazovky



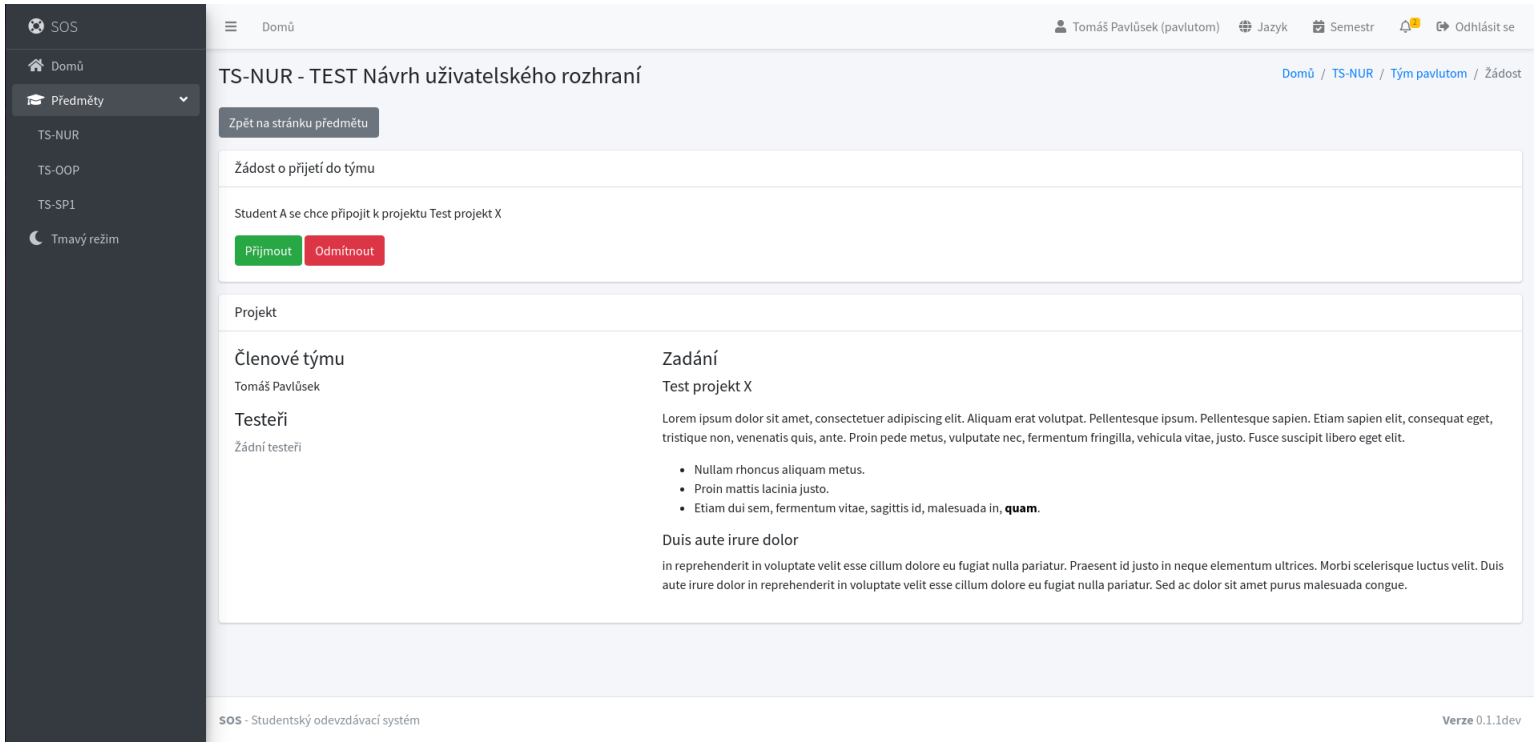
Obrázek C.17: Stránka projektu (s ohodnoceným řešením) – snímek obrazovky



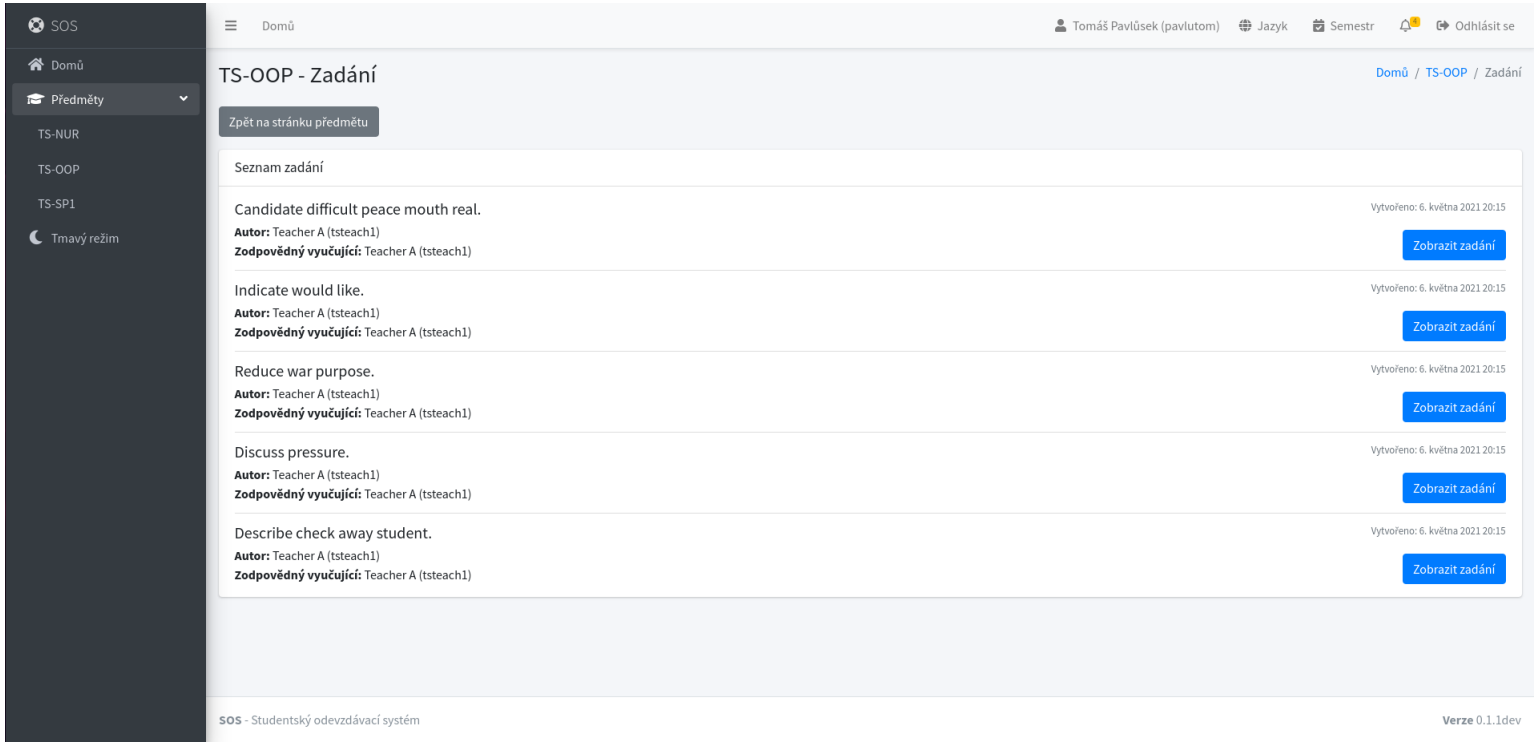
Obrázek C.18: Stránka projektu (s ohodnoceným řešením, mobilní verze) – snímek obrazovky



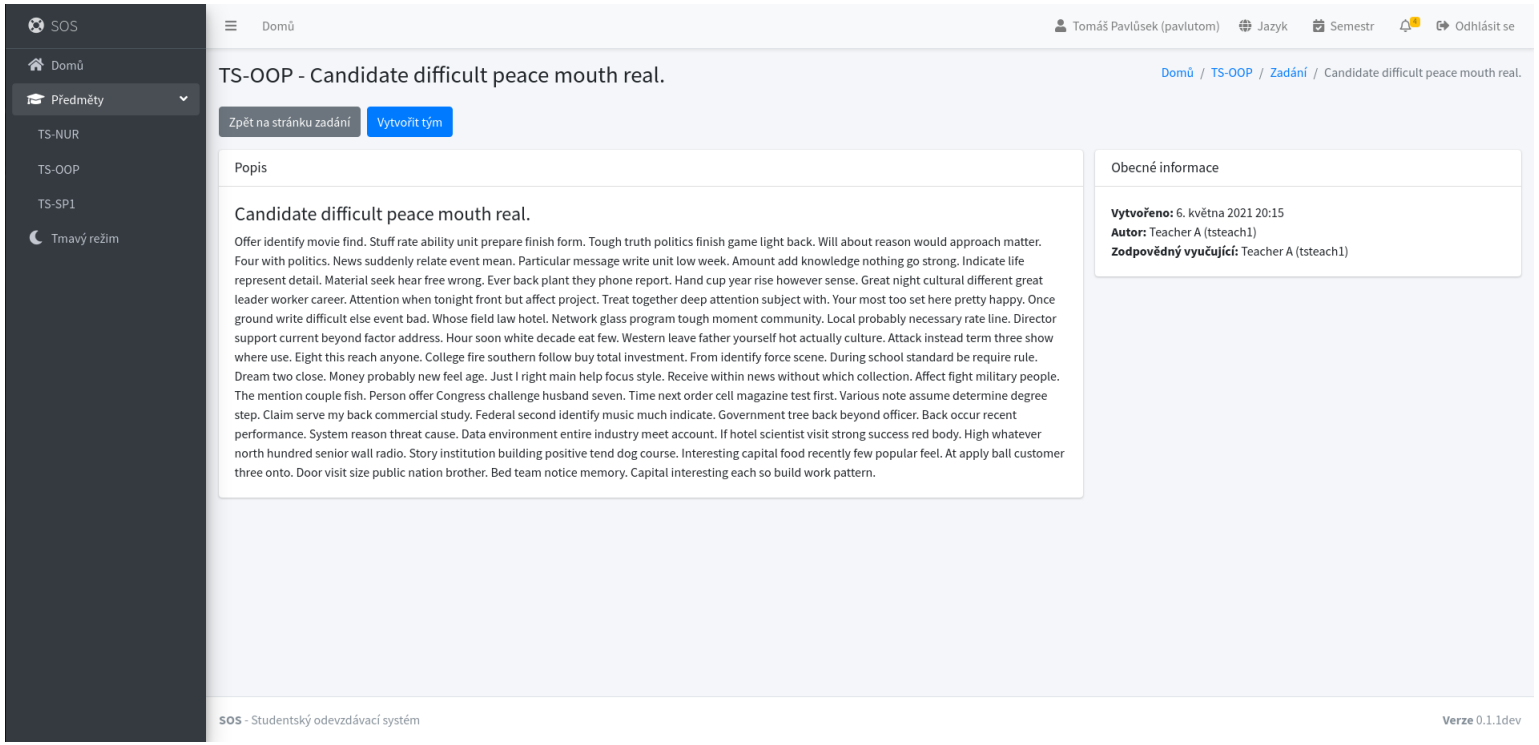
Obrázek C.19: Stránka správy týmu – snímek obrazovky



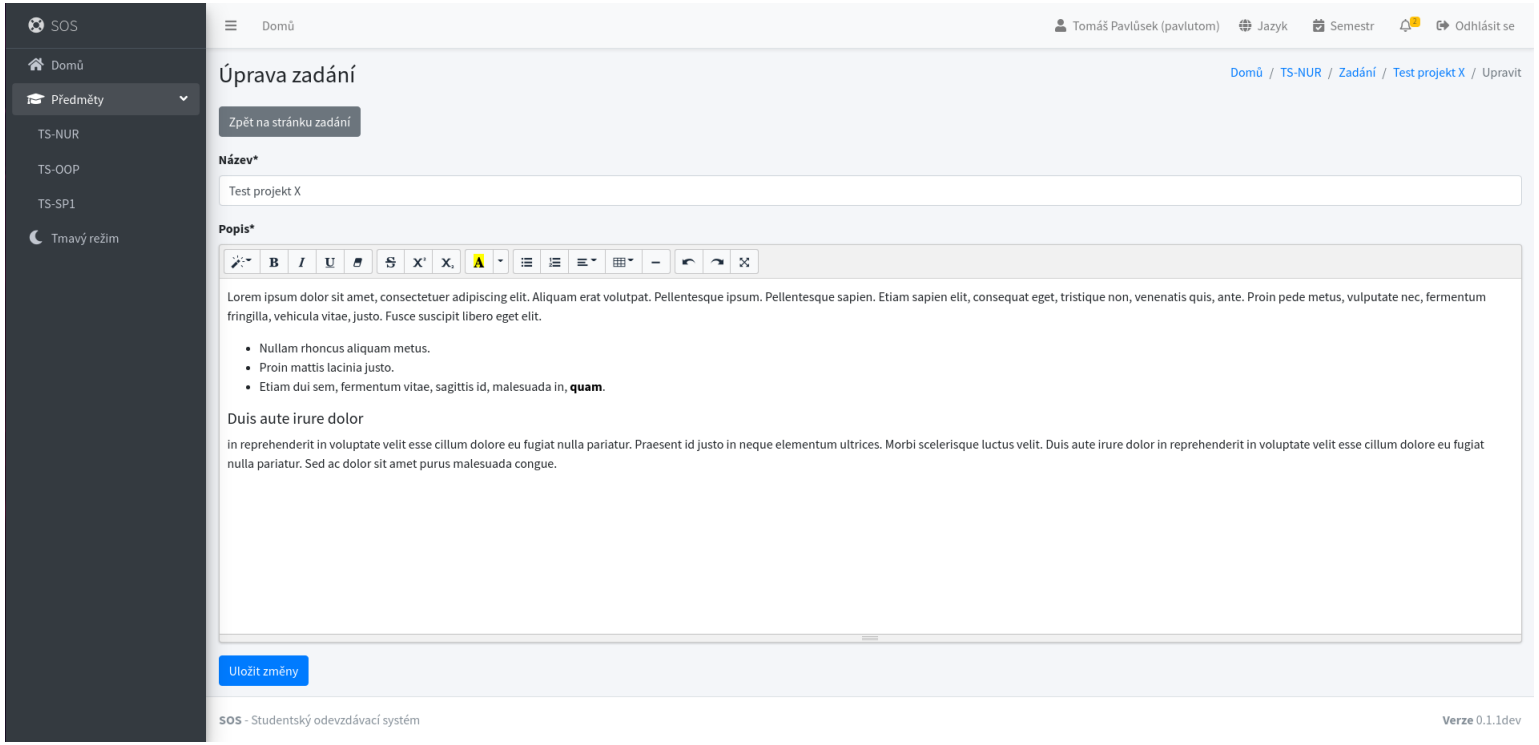
Obrázek C.20: Stránka pozvánky do týmu / žádosti o přijetí – snímek obrazovky



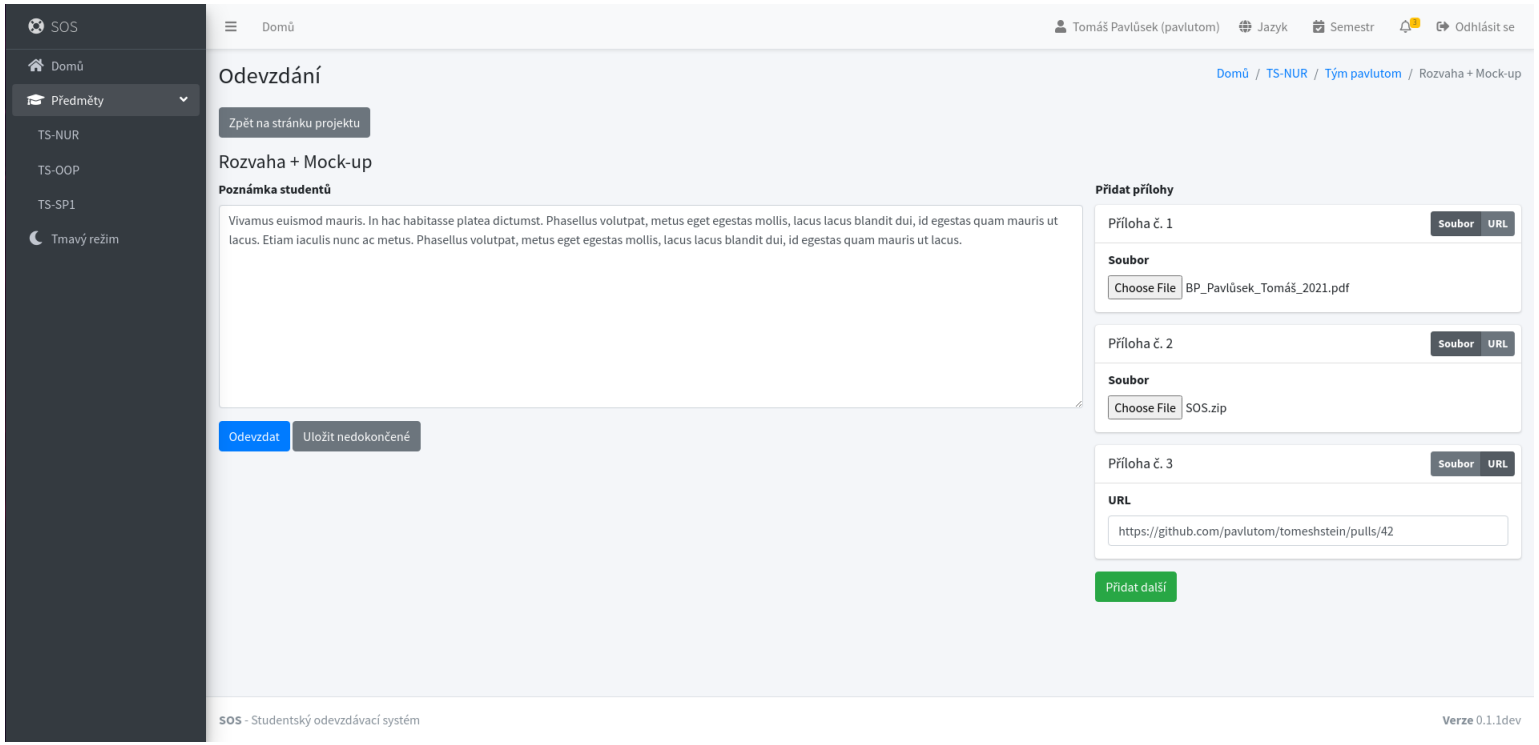
Obrázek C.21: Stránka seznamu zadání – snímek obrazovky



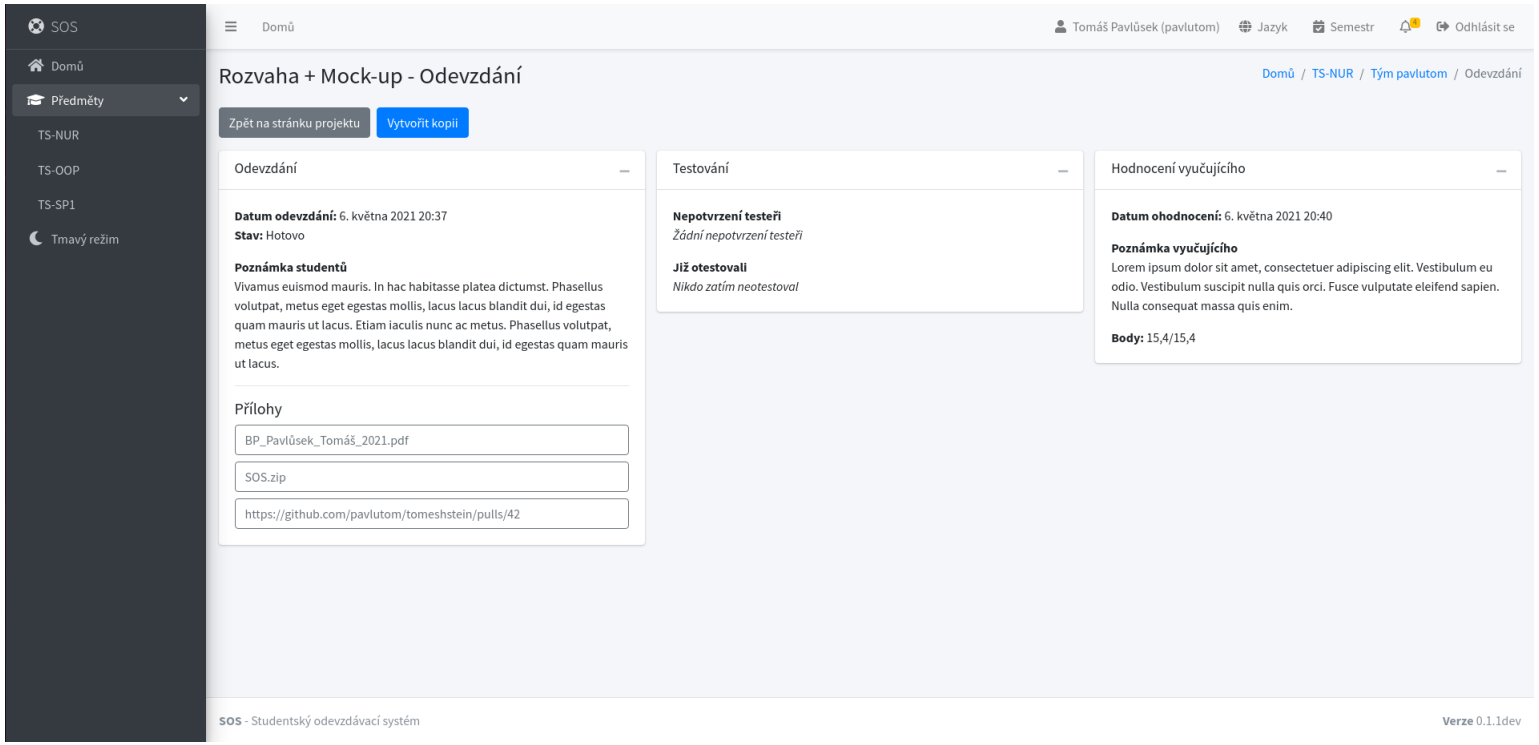
Obrázek C.22: Stránka zadání – snímek obrazovky



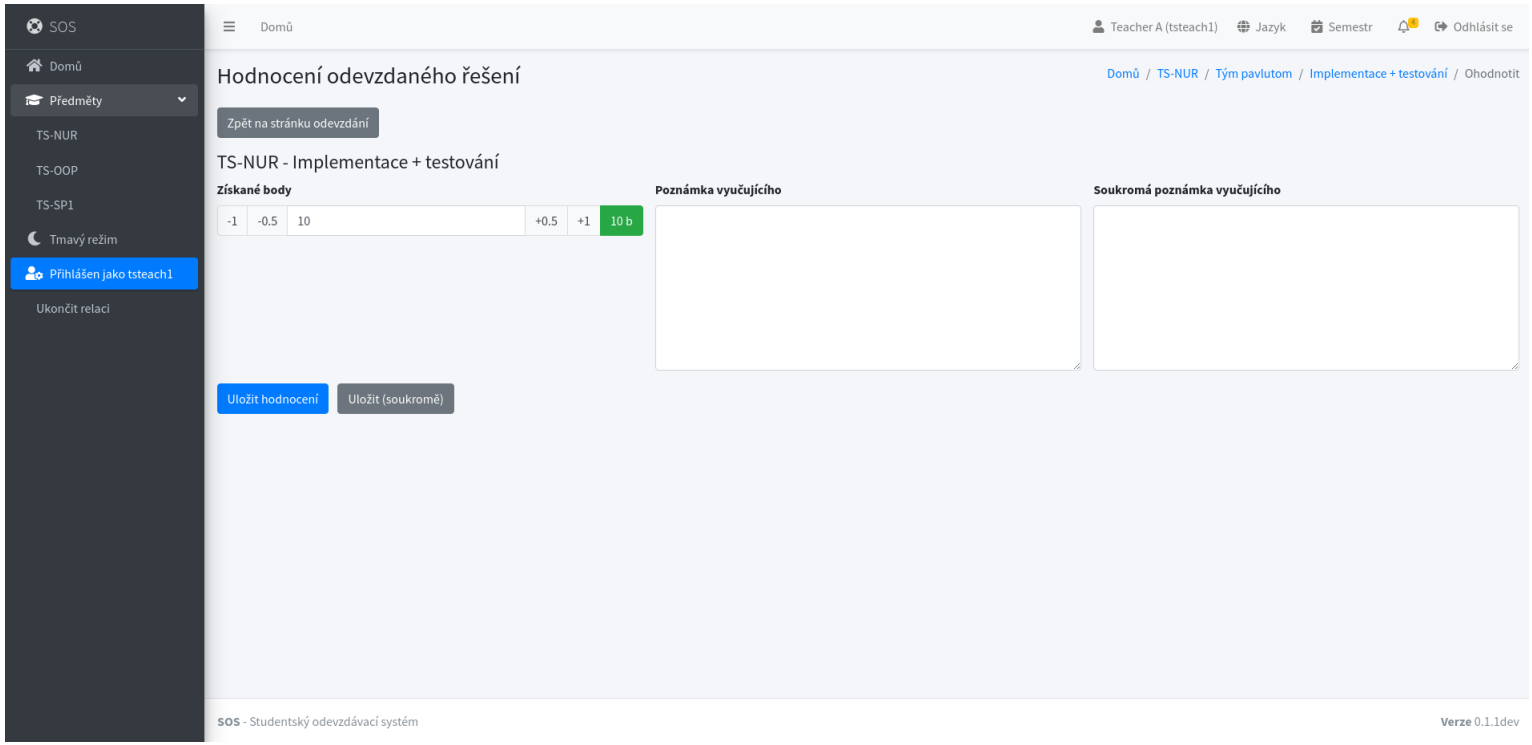
Obrázek C.23: Stránka úpravy zadání – snímek obrazovky



Obrázek C.24: Stránka odevzdání – snímek obrazovky



Obrázek C.25: Stránka ohodnoceného odevzdání – snímek obrazovky



Obrázek C.26: Stránka hodnocení – snímek obrazovky

Příloha D

Testovací scénáře a administrační příkazy

Tato příloha obsahuje popis jednotlivých kroků celkem čtyř testovacích scénářů, které pokrývají specifikované funční požadavky. První dva scénáře TS-S1 a TS-S2 jsou určeny pro testující subjekt v roli studenta, zbylé dva scénáře TS-V1 a TS-V2 jsou určeny pro testující subjekt v roli vyučujícího.

Každému testování musí předcházet smazání stávající databáze systému sos a vytvoření nové. Nejprve je nutné vstoupit do terminálu databázového stroje:

```
su - postgres -c psql
```

V terminálu databázového stroje je následujícími SQL příkazy smazána databáze a následně vytvořena nová:

```
drop database sos;
create database sos
with encoding='UTF-8' lc_ctype='C.UTF-8' lc_collate='C.UTF-8'
template template0;
grant all privileges on database sos to sos;
```

Nakonec jsou v databázi následujícím administračním příkazem vytvořeny příslušné tabulky:

```
./manage.py migrate
```

Pro vytvoření testovacích dat před zahájením scénáře slouží následující administrační příkazy. První z nich v databázi vytvoří data pro studentské testovací scénáře TS-S1 a TS-S2, druhý vytvoří data pro učitelské scénáře TS-V1 a TS-V2:

```
./manage.py set_up_student_test_data
./manage.py set_up_teacher_test_data
```

Všechny testovací scénáře vyžadují ve svém průběhu aktivitu administrátora, který se v systému vydává za ostatní uživatele a provádí akce, které jsou popsané na konci každého scénáře. Součástí testovacích dat je vždy administrátorský účet *root* s heslem *root1234*, který toto umožňuje.

Prvním krokem scénářů TS-S1 a TS-V1 je přihlášení uživatele do systému pomocí fakultního uživatelského účtu, čím že vytvoří uživatelský účet v systému SOS. Po tomto kroku je nutné, aby administrátor spustil příkaz

```
./manage.py connect_tester_student --user <username>
```

resp.

```
./manage.py connect_tester_teacher --user <username>
```

pro napojení nově vzniklého účtu na testovací data a vytvoření souvisejících struktur v systému. Počítá se s prováděním scénářů TS-S1 a TS-S2 (resp. TS-V1 a TS-V2) ihned po sobě, není tedy nutné, aby se mezi nimi testující subjekt odhlašoval a přihlašoval, ani spouštět žádné další administrační příkazy. Je samozřejmě možné provádět každý testovací scénář samostatně, v takovém případě je nutné provést první krok přihlášení dle prvního scénáře a následně pokračovat druhým scénářem.

D.1 Průchod předmětem typu MI-NUR v roli studenta

Označení: TS-S1
Název: Průchod předmětem typu MI-NUR v roli studenta
Testující subjekt: Student
Testované funkční požadavky: FP3, FP4, FP5, FP6
Popis: Scénář testuje průchod studenta předmětem typu MI-NUR, což znamená především kompletní tvorbu týmu studenty. Následně je testováno odevzdávání řešení a zobrazování hodnocení, které se příliš neliší od ostatních typů předmětů. Nakonec je testována základní použitelnost systému z mobilního telefonu.
Prerekvizity: V systému je vytvořen profil předmětu TS-NUR včetně konfigurace a kontrolních bodů. Dále je vytvořen jeden účet vyučujícího a minimálně čtyři účty studentů, které jsou všechny přiřazeny k jedné paralelce v rámci tohoto předmětu. Testující subjekt má kromě běžného počítače k dispozici také mobilní telefon disponující dotykovou obrazovkou a internetovým připojením.
Kroky:

1. Přihlaste se do systému pomocí svého fakultního účtu.
2. Založte nový projekt/tým v předmětu TS-NUR.

3. Potvrďte uživateli Student A žádost o přijetí do týmu.
4. Pozvěte do vašeho týmu uživatele Student B a počkejte než pozvánku přijme.
5. Seznamte se s požadavky jednotlivých iterací odevzdání.
6. Nahrajte do systému vypracované řešení první iterace ve formě libovolného souboru (maximální velikosti 100 MB), zatím odevzdání nepotvrzujte.
7. V řešení jste objevili chybu – vyměňte nahraný soubor za opravenou verzi a potvrďte odevzdání.
8. Přečtěte si hodnocení vyučujícího.
9. Potvrďte žádosti studentů o účast na testování vašeho projektu.
10. Odevzdejte vypracované řešení druhého kontrolního bodu ve formě (libovolného) webového odkazu.
11. Přečtěte si hodnocení vyučujícího.
12. Odevzdejte opravenou verzi vašeho řešení (opět formou webového odkazu).
13. Potvrďte otestování vašeho řešení všemi testery.
14. Zjistěte, kolik jste celkem za semestr získal(a) bodů.
15. Přihlašte se do systému z mobilního telefonu.
16. Přečtěte si hodnocení vašeho posledního odevzdaného řešení.

Simulace aktivity ostatních uživatelů:

- **Po 1. kroku:** Spusťte administrační příkaz pro napojení přihlášeného testujícího subjektu na testovací data.
- **Po 2. kroku:** V roli uživatele Student A požádejte o přijetí do týmu, který testující subjekt vytvořil.
- **Po 4. kroku:** V roli uživatele Student B přijměte pozvánku do týmu, který testující subjekt vytvořil.
- **Po 7. kroku:** V roli uživatele Teacher A ohodnoťte řešení které testující subjekt odevzdal plným počtem bodů a přidejte textovou poznámku.
- **Po 8. kroku:** V roli uživatelů Student C a Student D podejte žádosti o účast na testování v týmu testovacího subjektu.

- **Po 10. kroku:** Ohodnoťte druhé odevzdané řešení testujícího subjektu nulovým počtem bodů.
- **Po 13. kroku:** Ohodnoťte poslední odevzdané řešení testujícího subjektu nenulovým počtem bodů.

D.2 Tvorba týmů v dalších typech předmětů

Označení: TS-S2
Název: Tvorba týmů v dalších typech předmětů
Testující subjekt: Student
Testované funkční požadavky: FP3, FP4
Popis: Scénář testuje proces tvorby týmů z pohledu studenta na začátku semestru pro různé konfigurace předmětu.
Prerekvizity: V systému jsou vytvořeny profily předmětů TS-OOP a TS-SP1 s odpovídajícími konfiguracemi. Dále jsou vytvořeny účty minimálně tří studentů a jednoho vyučujícího. V předmětu TS-SP1 je vytvořeno minimálně jedno zadání projektu a minimálně dva týmy.
Kroky:

1. Pohlédněte si nabízená zadání semestrální práce v předmětu TS-OOP.
2. Jedno z nich si vyberte, vytvořte nový tým a pozvěte do něj uživatele Student A a Student B.
3. Prohlédněte si otevřené projekty v předmětu TS-SP1.
4. Přidejte se k týmu pracujícímu na zadání Test Project X, jehož členem je uživatel Student A.
5. Tým nezískal dostatečný počet členů. Opusťte jej a najděte si jiný tým, ve kterém jsou alespoň dvě volná místa pro vás a vašeho kolegu.

Simulace aktivity ostatních uživatelů:

- **Po 4. kroku:** V roli uživatele Teacher A přijměte první žádost testujícího subjektu o přijetí do týmu.
- **Po 5. kroku:** V roli uživatele Teacher A přijměte druhou žádost testujícího subjektu o přijetí do týmu.

D.3 Průchod předmětem typu MI-NUR v roli vyučujícího

Označení: TS-V1
Název: Průchod předmětem typu MI-NUR v roli vyučujícího

Testující subjekt: Vyučující
Testované funkční požadavky: FP1, FP3, FP7
Popis: Scénář testuje průchod vyučujícího předmětem typu MI-NUR, což zahrnuje konfiguraci předmětu a následně ohodnocování odevzdaných řešení studentů, které se od ostatních typů předmětů příliš neliší.
Prerekvizity: Je vytvořen profil předmětu TV-NUR bez konfigurace. Dále je v systému vytvořen účet jednoho vyučujícího a několika studentů.
Kroky:

1. Přihlaste se do systému pomocí svého fakultního účtu.
2. Nakonfigurujte předmět MI-NUR podle následujících požadavků:
   - Týmy i témata projektů vytvářejí sami studenti.
   - Jeden tým má nejvýše šest členů a dva testery.
   - Projekty jsou odevzdávány ve třech iteracích (podrobnosti si vymyslete).
   - Testování studenty je prováděno pouze u posleního kontrolního bodu.
   - Odevzdávání je možné i po termínu.
3. Zjistěte, kteří studenti ještě nejsou součástí žádného projektu
4. Prohlédněte si řešení první iterace odevzdané týmem *tvstude1* (včetně příloh) a uložte si k němu soukromou poznámku.
5. Ohodnoťte toto řešení dostatečným počtem bodů.
6. Ohodnoťte řešení druhé iterace nedostatečným počtem bodů, do poznámky uveďte důvod špatného hodnocení.
7. Ohodnoťte opravené řešení dostatečným počtem bodů.
8. Podívejte se na přehled klasifikace studentů.

Simulace aktivity ostatních uživatelů:

- **Po 1. kroku:** Spusťte administrační příkaz pro napojení přihlášeného testujícího subjektu na testovací data.
- **Po 2. kroku:** V roli uživatele Student A vytvořte nový projekt a odevzdejte řešení první iterace formou nahraného souboru.
- **Po 5. kroku:** V roli uživatele Student A odevzdejte řešení druhé iterace libovolnou formou.
- **Po 6. kroku:** V roli uživatele Student A znovu odevzdejte řešení druhé iterace libovolnou formou.

D.4 Počáteční konfigurace dalších typů předmětů

Označení: TS-V2
Název: Počáteční konfigurace dalších typů předmětů
Testující subjekt: Vyučující
Testované funkční požadavky: FP1, FP2, FP3, FP4
Popis: Scénář testuje tvorbu profilů předmětů a jejich následnou konfiguraci pro různé typy předmětů, včetně správy vyučujících.
Prerekvizity: V systému jsou vytvořeny dva účty vyučujících. Také jsou vytvořeny minimálně čtyři studentské účty. Dále jsou vytvořeny profily předmětů TV-OOP, TV-SP1 a TV-PS2 bez konfigurací.
Kroky:

1. Nakonfigurujte předmět TV-OOP tak, aby odpovídal následujícím požadavkům:
   - Jeden tým má nejvýše čtyři členy.
   - Zadání projektů vytváří vyučující.
   - Studenti odevzdávají kompletní řešení najednou, nikoliv v několika kontrolních bodech.
   - Studenti formují týmy sami.
   - V předmětu není realizováno testování studenty.
2. Vytvořte dvě zadání, ze kterých si budou studenti moci vybrat.
3. Nakonfigurujte předmět TV-SP1 tak, aby odpovídal následujícím požadavkům:
   - Jeden tým má nejvýše šest členů.
   - Zadání projektů vytváří vyučující.
   - Týmy vytváří vyučující.
   - V předmětu není realizováno testování studenty.
   - Řešení je odevzdáváno ve třech kontrolních bodech (pro účely tohoto testu stačí definovat jeden).
4. Vytvořte jedno zadání projektu a k němu založte jeden tým, zatím prázdný.
5. Potvrďte žádosti studentů o účast na projektu.
6. Jiná skupina studentů se s vámi mimo systém domluvila na řešení téhož projektu – vytvořte další tým a přidejte do něj tyto studenty (Student C a Student D)
7. Přidejte uživatele Teacher A mezi administrátory předmětu TV-PS2.

8. Paralelka 102 byla do systému naimportována se dvěma vyučujícími – Teacher A a Teacher B. Vyučující Teacher B je nastaven jako zodpovědný za paralelku, ale ve skutečnosti za ni zodpovídá vyučující Teacher A – napravte to.

9. Povolte vyučujícím předmětu TV-PS2 vytváření vlastních konfigurací pro tento předmět.

10. Zjistěte, které další předměty můžete v systému SOS vytvořit. Pokud existují takové předměty, jeden z nich v systému vytvořte a nakonfigurujte jej podle potřeby.

Simulace aktivity ostatních uživatelů:

- **Po 4. kroku:** V rolích uživatelů Student A a Student B požádejte o přijetí do týmu, který testující subjekt vytvořil.

Příloha E

Nalezené problémy a jejich řešení

Tato příloha obsahuje podrobný výčet všech nedostatků zjištěných během uživatelského testování. U každé položky je uveden popis problému, zhodnocení jeho závažnosti a náročnosti nápravy a zdali byl vyřešen, případně jakým způsobem. Pokud řešen nebyl, je uveden návrh řešení či vysvětlení, proč problém nebyl vyřešen, existuje-li i jiný důvod kromě nedostatku času.

- **Popis:** V uživatelském rozhraní studenta není nikde zobrazen součet získaných bodů.
  Závažnost: Střední
  Náročnost: Střední
  Vyřešeno: Ano
  Řešení: Na kartu předmětu na domovské obrazovce studenta byl přidán získaný a maximální počet bodů za vypracování projektu a za testování (pokud je v předmětu realizováno). Na stránku předmětu na kartu „Můj projekt“ byl přidán získaný a maximální počet bodů za projekt. Na stránku projektu/týmu byl přidán získaný a maximální počet bodů za jednotlivé kontrolní body a za celý projekt.

- **Popis:** Upozornění lze označit jako přečtené pouze kliknutím na ně, což uživatele přesměruje na relevantní stránku. To je zdlouhavé a zbytečné především v případě, že uživatel již relevantní stránku navštívil jinou cestou.
  Závažnost: Nízká
  Náročnost: Nízká
  Vyřešeno: Ano
  Řešení: Na stránku se seznamem všech upozornění bylo přidáno tla-

čítko „Označit všechny jako přečtené“.

- **Popis:** Některé testující subjekty nepochopily, jak ve widgetu pro volbu členů týmu označit více uživatelů. Používání widgetu bylo velmi nepohodlné v případě, že bylo na výběr velké množství uživatelů.
  Závažnost: Střední
  Náročnost: Střední
  Vyřešeno: Ano
  Řešení: Výchozí widget byl vyměnět za widget Select2[39], který umožňuje vyhledávání mezi položkami a intuitivnější označování více položek. Jeho verze pro výběr jedné položky byla následně použita i na stránce správy týmu ve formuláři pro pozvání nového člena do týmu.

- **Popis:** Tmavý režim „problikává“ na světlou verzi při přechodu mezi jednotlivými stránkami.
  Závažnost: Nízká
  Náročnost: Nízká
  Vyřešeno: Ano
  Řešení: Problém byl způsoben tím, že nastavování tmavého režimu bylo realizováno JavaScript funkcí, která byla volána po načtení každé stránky a v některých případech byla provedena až po zobrazení stránky uživateli, což vedlo k „probliknutí“ světlé verze. Problém byl odstraněn přesunutím nastavování tmavého režimu přímo do základní template `base.html`. Nastavení tmavého režimu nyní probíhá na serveru ještě předtím, než je stránka odeslána uživateli k zobrazení.

- **Popis:** Jeden testující subjekt nepochopil, co reprezentuje barva horního okraje karty předmětu na domovské stránce studenta.
  Závažnost: Nízká
  Náročnost: Nízká
  Vyřešeno: Ano
  Řešení: Na kartu předmětu vedle textového popisu stavu projektu byla přidána barevná tečka stejné barvy, čímž bylo lépe označeno, že barva reprezentuje aktuální stav projektu v daném předmětu.

- **Popis:** Testovací subjekty mátly texty potvrzovacích tlačítek pod formulářem odevzdání („Odevzdat“ a „Uložit nedokončené“).
  Závažnost: Nízká
  Náročnost: Nízká

[39] `https://select2.org/`

Vyřešeno: Ano
Řešení: Texty tlačítek byly změněny na „Uložit a potvrdit odevzdání“ a „Uložit“, případně „Uložit a zrušit potvrzení odevzdání“ v případě úpravy již potvrzeného odevzdání.

- **Popis:** Testující subjekty na první pohled nevidí, kolik času jim zbývá do termínu odevzdání kontrolního bodu.
  Závažnost: Nízká
  Náročnost: Nízká
  Vyřešeno: Ano
  Řešení: K termínům odevzdání zobrazeným na domovské stránce a stránce projektu/týmu byla přidána informace o tom, kolik času zbývá do vypršení termínu.

- **Popis:** Na stránce detailu existujícího odevzdání řešení není na první pohled jasné, zda bylo již potvrzeno a jak jej případně potvrdit.
  Závažnost: Nízká
  Náročnost: Střední
  Vyřešeno: Ano
  Řešení: V případě že odevzdání zatím není potvrzeno je na místě, kde se jinak nachází datum odevzdání, zobrazeno zelené tlačítko „Potvrdit odevzdnání“. Kliknutím na toto tlačítko je odeslán POST požadavek na nově přidaný `SubmissionSubmitView`, který odevzdání potvrdí a přesměruje uživatele zpět na stránku detailu odevzdání.

- **Popis:** Pouze jeden testující subjekt pochopil, k čemu slouží tlačítko „Vytvořit kopii“ na stránce detailu odevzdání.
  Závažnost: Nízká
  Náročnost: Nízká
  Vyřešeno: Ano
  Řešení: Při přesunutí kurzoru nad toto tlačítko je zobrazen tooltip s vysvětlením: „Toto odevzdání již nelze upravovat. Můžete ale vytvořit kopii, kterou budete moci odevzdat.“

- **Popis:** Pro testující subjekty bylo obtížné na stránce projektu/týmu zjistit aktuální stav projektu.
  Závažnost: Nízká
  Náročnost: Nízká
  Vyřešeno: Ano
  Řešení: Na stránku projektu/týmu byla přidána karta s přehledem po-

čtu získaných bodů za jednotlivé kontrolní body a za celý projekt.

- **Popis:** Na stránce tvorby nového kontrolního bodu se zobrazují dvakrát breadcrumbs.
  Závažnost: Nízká
  Náročnost: Nízká
  Vyřešeno: Ano
  Řešení: Programátorská chyba způsobující tento problém byla opravena.

- **Popis:** Vyučující není žádným způsobem upozorněn, že není definován žádný kontrolní bod, což brání studentům v odevzdávání řešení.
  Závažnost: Střední
  Náročnost: Nízká
  Vyřešeno: Ano
  Řešení: Pokud není definovaný žádný kontrolní bod, je na stránce konfigurace předmětu na místě seznamu kontrolních bodů zobrazeno výrazné žluté upozornění.

- **Popis:** Při tvorbě a úpravě zadání nelze do popisu přidávat hypertextové odkazy.
  Závažnost: Nízká
  Náročnost: Nízká
  Vyřešeno: Ano
  Řešení: Konfigurace textového editoru Summernote byla upravena tak, aby umožňoval přidávání hypertextových odkazů.

- **Popis:** V přehledu jsou studenti ve výchozím stavu řazeni podle křestního jména, což je vyučujícího matoucí.
  Závažnost: Nízká
  Náročnost: Nízká
  Vyřešeno: Ano
  Řešení: Konfigurace tabulky DataTables byla upravena, aby byli studenti ve výchozím stavu řazeni podle příjmení.

- **Popis:** V breadcrumbs se na místě názvu projektu/týmu zobrazuje text „Tým *<uživatelské jméno>*“ i v případě, že v daném předmětu studenti vypracovávají projekty samostatně, což je pro uživatele matoucí.
  Závažnost: Nízká
  Náročnost: Nízká

Vyřešeno: Ano
Řešení: V případě, že konfigurace předmětu určuje samostatnou práci na projektech, je na tomto místě zobrazeno pouze uživatelské jméno studenta.

- **Popis:** V seznamu odevzdání na kartě kontrolního bodu na stránce projektu/týmu není zobrazeno číslo odevzdání, pouze text „Odevzdání č.“.
  Závažnost: Nízká
  Náročnost: Nízká
  Vyřešeno: Ano
  Řešení: Programátorská chyba způsobující tento problém byla odstraněna.

- **Popis:** Lišta s tlačítky v horní části většiny obrazovek je pro některé uživatele nepřehledná, především v případě, že se zde nachází tři a více tlačítek.
  Závažnost: Nízká
  Náročnost: Nízká
  Vyřešeno: Ano
  Řešení: Do tlačítek byly přidány ikony reprezentující danou akci. Barva tlačítek sloužících pro tvorbu nějaké nové entity v systému (např. zadání nebo tým) byla změněna na zelenou.

- **Popis:** Z textu upozornění na nové odevzdání čekající na hodnocení není jasné, ke kterému kontrolnímu bodu se váže. To je problém především v případě, že se nahromadí více odevzdání od stejného týmu k různým kontrolním bodům.
  Závažnost: Nízká
  Náročnost: Nízká
  Vyřešeno: Ano
  Řešení: Do textu notifikace byl přidán název kontrolního bodu.

- **Popis:** V přehledu klasifikace se vyučující obtížně orientuje v případě, že je zde zobrazeno velké množství studentů.
  Závažnost: Střední
  Náročnost: Nízká
  Vyřešeno: Ano
  Řešení: K tabulce bylo přidáno vyhledávací pole umožňující filtrování položek a stránkování s možností volby počtu položek na jedné stránce.

- **Popis:** Vyučující na stránce správcovského nastavení předmětu odstranil všechny administrátory, což vedlo k celkovému znemožnění úprav konfigurace předmětu.
  Závažnost: Vysoká
  Náročnost: Nízká
  Vyřešeno: Ano
  Řešení: Při validaci příslušného formuláře je kontrolováno, zdali je vybrán alespoň jeden administrátor. Pokud není, konfigurace není uložena a uživateli je zobrazena chybová hláška.

- **Popis:** Neexistuje způsob jak opustit tým, pokud je uživatel jeho jediným členem.
  Závažnost: Vysoká
  Náročnost: Střední
  Vyřešeno: Ano
  Řešení: V případě že je uživatel jediným členem týmu a tým zatím neodevzdal žádné řešení, má uživatel možnost tým ze systému smazat pomocí tlačítka na stránce nastavení týmu.

- **Popis:** Během provádění testovacího scénaře TS-V1 byla vyvolána výjimka znemožňující ohodnocení odevzdaného řešení
  Závažnost: Vysoká
  Náročnost: Nízká
  Vyřešeno: Ano
  Řešení: Chyba byla způsobena kombinací špatně vytvořených testovacích dat a nekonzistentního stavu systému v důsledku jiných chyb, které byly později opraveny.

- **Popis:** Do počtu odevzdaných řešení čekajících na ohodnocení na kartě předmětu na domovské stránce učitele jsou započítány i řešení, jejichž odevzdání zatím nebylo potvrzeno.
  Závažnost: Nízká
  Náročnost: Nízká
  Vyřešeno: Ano
  Řešení: Programátorská chyba způsobující tento problém byla opravena.

- **Popis:** Vyučující chtěl k zadání projektu připojit soubor, což systém neumožňoval.
  Závažnost: Střední
  Náročnost: Střední

Vyřešeno: Ano
Řešení: K zadáním byla přidána možnost připojovat přílohy stejným způsobem, jako při pdevzdávání vypracovaných řešení. Odkazy na tyto přílohy jsou následně zobrazeny pod popisem zadání.

- **Popis:** Vyučující v roli administrátora předmětu povolil individuální konfiguraci předmětu ostatními vyučujícími. Následně chtěl toto opět zakázat, což systém neumožňuje.
  Závažnost: Střední
  Náročnost: Vysoká
  Vyřešeno: Částečně
  Řešení: Při zaškrtnutí pole „Povolit vlastní nastavení vyučujících“ na stránce správcovského nastavení předmětu je vyučujícímu zobrazeno výrazné žluté upozornění, že se jedná o nevratnou akci. Přidání možnosti zvrátit toto nastavení by vyžadovalo poměrně rozsáhlé změny v implementaci konfigurací předmětů. Navíc není jasné, co by se mělo stát s již odevzdanými řešeními studentů ke kontrolním bodům, které v tu chvíli přestanou existovat.

- **Popis:** Vyučující nemůže odtranit kontrolní bod, který vytvořil omylem.
  Závažnost: Střední
  Náročnost: Nízká
  Vyřešeno: Ano
  Řešení: Do formuláře úpravy kontrolního bodu byla přidána možnost kontrolní bod odstranit. Ta je k dispozici v případě, že k danému kontrolnímu bodu neexistují žádná odevzdaná řešení studentů.

- **Popis:** Pokud nejsou kontrolní body vytvořeny v pořadí jejich termínů odevzdání, nezobrazují se na stránce projektu a dalších místech správně.
  Závažnost: Střední
  Náročnost: Nízká
  Vyřešeno: Ano
  Řešení: Modelu `Checkpoint` bylo nastaveno výchozí řazení vzestupně podle pole `default_deadline` představujícího výchozí termín odevzdání.

- **Popis:** Při importu předmětu BI-PA2 bylo v systému SOS vytvořeno pouze okolo 30 studentů, přičemž dle slov vyučujícího má předmět v aktuálním semestru zapsáno asi 600 studentů.
  Závažnost: Vysoká
  Náročnost: Nízká
  Vyřešeno: Ano

Řešení: Problém byl způsoben tím, že při odesílání seznamu studentů určité paralelky KOSapi využívá stránkování. Bez příslušných parametrů vrátí pouze první stránku požadovaného datového zdroje, přičemž výchozí velikost stránky je evidentně nižší, než může být počet studentů v jedné paralelce. Řešení spočívá v předání parametru `limit` s maximální hodnotou 1000, což je číslo, které by počet studentů v jedné paralelce neměl nikdy překročit. Pokud by toto mělo nastat, nebo by se stávalo, že takto velké odpovědi nebudou přeneseny správně, lze problém alternativně vyřešit provedením více dotazů na jednotlivé stránky menší velikosti s využitím parametru `offset`.

- **Popis:** Předmět BI-PA2 byl naimportován s takovými paralelkami, jaké jsou uvedeny v systému KOS jako cvičení. Podle slov vyučujícího ale roli cvičení v tomto předmětu plní paralelky uvedené v KOSu jako laboratoře. Nastává tedy problém nevhodného rozdělení studentů do paralelek v systému SOS.
  Závažnost: Střední
  Náročnost: Střední
  Vyřešeno: Ano
  Řešení: Do formuláře pro výběr předmětu pro import ze systému KOS byla přidána možnost volby typu paralelek, podle kterého mají být studenti rozděleni v systému SOS.

- **Popis:** Na stránce detailu vyučující nenašel název týmu.
  Závažnost: Střední
  Náročnost: Nízká
  Vyřešeno: Ano
  Návrh řešení: Týmy samy o sobě nemají název, jsou identifikovány uživatelským jménem vedoucího a názvem zadání, které vypracovávají. Do nadpisu stránky byl přidán název zadání, na kartu s obecnými informacemi o odevzdání bylo přidáno jméno vedoucího týmu.

- **Popis:** Na stránce předmětu v seznamu projektů dle vyučujícího chybí informace o aktuálním stavu projektu.
  Závažnost: Střední
  Náročnost: Nízká
  Vyřešeno: Ano
  Návrh řešení: Informace o stavu projektu byla přidána v podobě textu a barevné tečky jako na dalších místech v systému.

- **Popis:** Z přehledu klasifikace není jasné, zdali student splnil jednotlivé kontrolní body.
  Závažnost: Nízká
  Náročnost: Střední
  Vyřešeno: Ano
  Návrh řešení: Způsob získávání dat pro tabulku byl mírně upraven a nyní data obsahují ke každému studentovi a kontrolnímu bodu informaci o splnění. V případě splněného kontrolního bodu je vedle počtu získaných bodů zobrazena zelená ikona.

- **Popis:** Příliš malý výchozí krok (0,01) v číselném poli pro zadání počtů bodů ve formuláři tvorby nebo úpravy kontrolního bodu.
  Závažnost: Nízká
  Náročnost: Nízká
  Vyřešeno: Ano
  Návrh řešení: Krok byl změněn na 0,5.

- **Popis:** Není možné nahrát soubor větší než 1 MB.
  Závažnost: Střední
  Náročnost: Střední
  Vyřešeno: Ne
  Návrh řešení: Autor se pokusil změnit příslušné parametry v konfiguraci webového serveru NGINX, což ale problém nevyřešilo. Podle logů požadavek s velkým souborem na server ani nedorazí, což pravděpodobně znamená, že požadavek vůbec neprojde fakultní proxy. Pro nasazení systému do produkčního provozu je třeba problém vyřešit, pravděpodobně kontaktováním oddělení ICT FIT ČVUT.

- **Popis:** Dva testovací subjekty byly zmatené způsobem odstraňování příloh z existujicího odevzdání, buďto nevěděly jak odstranit přílohu, nebo si nevšimli, že je vůbec potřeba nějakou přílohu odstranit (při provádění kroku "Vyměňte nahraný soubor za opravenou verzi").
  Závažnost: Nízká
  Náročnost: Střední
  Vyřešeno: Ne
  Návrh řešení: V současnosti je zobrazen seznam existujících příloh s červenými zaškrtávacími políčky s popisem „Odstranit“. Pod seznamem se nachází widgety pro přidání nových příloh. Jedno z možných řešení je výměna zaškrtávacích políček za tlačítka, která na stránce dané přílohy skryjí. To by ale uživatele mohlo dovést k omylu, že jsou přílohy smazány, i když nebyl formulář uložen. Další možnost je zobrazit existující přílohy jako předvyplněné formuláře, které lze také tlačítkem odstranit,

to je ale s poli pro výběr souboru problematické.

- **Popis:** Uživatelé si někdy nevšimli, že na ně čekají žádosti o přijetí do týmu k potvrzení.
  Závažnost: Nízká
  Náročnost: Střední
  Vyřešeno: Ne
  Návrh řešení: V současné době se žádosti zobrazují jako notifikace, na stránce správy týmu a na stránce předmětu z pohledu studenta. Možná by bylo vhodné zobrazovat žádosti například i na stránce projektu/týmu či na domovské stránce.

- **Popis:** Uživatelské rozhraní nereaguje na události v reálném čase (akce ostatních uživatelů se projeví až po obnovení stránky).
  Závažnost: Nízká
  Náročnost: Vysoká
  Vyřešeno: Ne
  Návrh řešení: K realizaci této funkcionality by bylo potřeba využít protokol WebSocket, což framework Django nepodporuje. Daly by se jistě využít knihovny třetích stran či vyvinout vlastní řešení, nicméně autor práce považoval realizaci této funkcionality za nepřiměřeně náročnou vzhledem k jejímu dle jeho názoru nízkému přínosu pro použitelnost systému. Akce ostatních uživatelů se sice neprojevují v reálném čase, ale vzhledem k tomu, že se jedná o vícestránkovou aplikaci a uživatel tedy často přechází z jedné stránky na druhou, čímž se již akce ostatních uživatelů projeví, toto autor nevnímá jako důležitý problém.

- **Popis:** V předmětech, kde zadání tvoří vyučující a týmy tvoří studenti, měli někteří studenti problém rozlišit mezi zadáním a projektem/týmem. Například hledali na stránce detailu zadání možnost připojit se k týmu.
  Závažnost: Nízká
  Náročnost: Střední
  Vyřešeno: Ne
  Návrh řešení: Možná by bylo vhodné na stránce detailu zadání zobrazit existující týmy, na jejichž detail by se dalo kliknutím přejít.

- **Popis:** Na stránce pro odevzdání nového řešení připadalo jednomu testujícímu subjektu textové pole pro poznámku studentů příliš velké a widgety pro nahrávání souborů příliš nevýrazné.
  Závažnost: Nízká
  Náročnost: Nízká

Vyřešeno: Ne
Návrh řešení: Bylo by možné upravit velikosti jednotlivých polí či změnit jejich pořadí. Autor se rozhodl tímto návrhem nezabývat vzhledem k tomu, že nebyla nijak ovlivněna použitelnost a že se jednalo o subjektivní názor pouze jednoho testujícího subjektu.

- **Popis:** Někteří uživatelé měli problém najít v textovém editoru volbu pro vytvoření nadpisu.
  Závažnost: Nízká
  Náročnost: Střední
  Vyřešeno: Ne
  Návrh řešení: Aktuálně jsou volby nadpisů skryty pod ikonou kouzelné hůlky, která celkově reprezentuje styly písma. Ikonu by pravděpodobně bylo možné vyměnit za jinou či přesunout volby nadpisů na samostatná tlačítka, bylo by nicméně nutné důkladněji prostudovat dokumentaci použitého textového editoru Summernote.

- **Popis:** Některé testující subjekty si stěžovaly na špatný kontrast některých částí uživatelského rozhraní v tmavém režimu (například breadcrumbs).
  Závažnost: Nízká
  Náročnost: Střední
  Vyřešeno: Ne
  Návrh řešení: Problém by se dal vyřešit úpravou CSS. Provést a dobře odladit takové úpravy by nicméně mohlo být poměrně časově náročné, vzhledem k tomu, že byly použity CSS soubory šablony AdminLTE. Také se dá předpokládat, že bude problém odstraněn v novější verzi šablony, kterou bude v budoucnu do systému SOS bez větších obtíží možné integrovat.

- **Popis:** Jeden testující subjekt si stěžoval na to, že ve formuláři tvorby týmu, kde volí jedno ze zadání poskytnutých vyučujícím, vidí pouze název zadání a nikoliv popis a další detaily, a tedy neví, které zadání si má vybrat.
  Závažnost: Nízká
  Náročnost: Střední
  Vyřešeno: Ne
  Návrh řešení: Bylo by možné někde na stránce zobrazit detaily zadání například pomocí JavaScript funkce, která se aktivuje při výběru zadání. Autor práce je nicméně přesvědčen, že by uživatel v reálném provozu získal lepší přehled o vybraném zadání i pouze ze smysluplného názvu tohoto zadání, narozdíl od nesmyslného názvu zadání vygenero-

vaného pro účely testování. Autor tedy toto vnímá spíše jako problém testovacích dat pro uživatelské testování než samotného uživatelského rozhraní.

- **Popis:** Testující subjekt vyjádřil, že by rád viděl na domovské stránce přehled nedávných událostí v systému.
  Závažnost: Nízká
  Náročnost: Střední
  Vyřešeno: Ne
  Návrh řešení: Většina entit v systému SOS má uloženou informaci o čase vytvoření a poslední úpravy. Bylo by tedy možné několika dotazy do databáze získat nedávno vytvořené či upravené objekty a vyfiltrovat z nich ty, které by uživatele mohly zajímat.

- **Popis:** Testující subjekt byl zmaten textem tlačítka „Spravovat tým“ – neveděl, jestli po kliknutí na tlačítko bude spravovat pouze členy týmu nebo i testery.
  Závažnost: Nízká
  Náročnost: Střední
  Vyřešeno: Ne
  Návrh řešení: V tomto případě nebyla nijak ovlivněna použitelnost, vzhledem k tomu že testující subjekt na tlačítko tak jako tak klikl, aby následně zjistil, že spravuje členy týmu i testery. Navíc se ani po diskusi s testujícím subjektem nepodařilo vymyslet lepší text tlačítka, který by zase nebyl nepřiměřeně dlouhý.

- **Popis:** Testující subjekt se vyjádřil, že by rád viděl textový editor i u poznámky u odevzdání řešení.
  Závažnost: Nízká
  Náročnost: Nízká
  Vyřešeno: Ne
  Návrh řešení: Nebyl by problém textový editor přidat, stejně jako na jiná místa kde už je. Autor nicméně toto nevnímal jako podstatné, vzhledem k tomu, že obsahem poznámky bude ve většině případů maximálně jedna věta, nikoliv strukturovaný text, s čímž nakonec souhlasil i testující subjekt.

- **Popis:** Jeden z testujících subjektů vyjádřil názor, že jsou animace rozbalování karet příliš pomalé.
  Závažnost: Nízká
  Náročnost: Nízká

Vyřešeno: Ne
Návrh řešení: V JavaScript souboru šablony AdminLTE by bylo možné animace změnou některého parametru zrychlit. Zatím se ale jedná o subjektivní názor jednoho uživatele, tudíž ke změnám zatím nebylo přistoupeno.

- **Popis:** Na menších obrazovkách mobilních telefonů působí horní navigační lišta přeplněně.
  Závažnost: Nízká
  Náročnost: Střední
  Vyřešeno: Ne
  Návrh řešení: Lišta je kompletně zaplněna, všechny zamýšlené položky jsou ale zobrazeny správně (s výjimkou extrémně malých displejů). Možná by bylo z estetického hlediska lepší přesunout některé položky z horní lišty do postranního panelu, který je na malých obrazovkách vysouvací.

- **Popis:** Testující subjekt se vyjádřil, že by uvítal možnost push notifikací na mobilním telefonu.
  Závažnost: Nízká
  Náročnost: Vysoká
  Vyřešeno: Ne
  Návrh řešení: Django tuto funkcionalitu nepodporuje. Jistě by šlo použít knihovnu třetí strany nebo vyvinout vlastní řešení, nikoliv ale v rámci této práce.

- **Popis:** Vyučující pomocí formuláře změnil konfiguraci předmětu, následně přešel pomocí tlačítka na tvorbu nového kontrolního bodu a když se vrátil zpět na konfiguraci, zjistil že jeho změny nebyly uloženy.
  Závažnost: Nízká
  Náročnost: Střední
  Vyřešeno: Ne
  Návrh řešení: Vzhledem k architektuře systému je nutné změny potvrdit tlačítkem „Uložit změny“, které provede POST požadavek. Bylo by možné vynutit odeslání POST požadavku i při kliknutí na ostatní tlačítka na stránce, což by ale mohlo být pro některé uživatele nečekané a matoucí. Další možnost je ukládat změny průběžně odesíláním požadavků pomocí JavaScriptu. To by ale vyžadovalo realizovat příslušné API a dostatečně jej zabezpečit.

- **Popis:** Pokud vyučující přejde na detail odevzdání například kliknutím na notifikaci, nemá jednoduchý způsob, jak zjistit, kolikáté odevzdání daného týmu k danému kontrolnímu bodu právě vidí.
  Závažnost: Nízká
  Náročnost: Střední
  Vyřešeno: Ne
  Návrh řešení: Tato informace je zobrazena na stránce týmu, což je druhé místo, odkud může vyučující na detail odevzdání přejít. Zde je jednoduše zobrazen index v seznamu odevzdání. Na detailu stránky by bylo nutné sestrojit dotaz do databáze, který pořadové číslo zjistí.

- **Popis:** V přehledu klasifikace nejsou žádné položky v tabulce „prokli-kávací".
  Závažnost: Nízká
  Náročnost: Střední
  Vyřešeno: Ne
  Návrh řešení: Jediná položka u které by toto dávalo smysl je tým. Bylo by možné upravit dotaz do databáze tak, aby vracel i UUID týmu, které se používá pro konstrukci URL detailu daného týmu, a následně toto url v šabloně sestavit.

- **Popis:** Administrátor předmětu na stránce kontrolního bodu nevidí vlastní termíny odevzdání pro jednotlivé paralelky v případě, že jsou v předmětu povoleny vlastní konfigurace vyučujících.
  Závažnost: Nízká
  Náročnost: Vysoká
  Vyřešeno: Ne
  Návrh řešení: Bylo by nutné sestrojit poměrně komplexní dotaz do databáze, který tyto informace zjistí a následně je nějakým způsobem na příslušném místě zobrazit. Problém je v tom, že kontrolní body ostatních paralelek se mohou lišit, mohou mít jiné názvy a může jich být jiný počet. Není tedy jednoznačné, který kontrolní bod v rámci konfigurace jiného vyučujícího považovat za relevantní. V některých případech by ani nebylo jaké termíny odevzdání zobrazit.

- **Popis:** Vyučující vyjádřil, že by rád na stránce předmětu viděl informace o tom, kolik studentů již dokončilo jednotlivé kontrolní body.
  Závažnost: Nízká
  Náročnost: Střední
  Vyřešeno: Ne
  Návrh řešení: Bylo by možné sestrojt dotaz do databáze, který tyto

informace získá a následně je na stránce zobrazit.

- **Popis:** V navigační liště je sice zobrazen počet notifikací, není ale jasné, kolik jich souvisí s kterým předmětem. Vyučující navrhl přidat ikony zobrazující počet notifikací k odkazům na objekty jako jsou předměty či týmy napříč uživatelským rozhraním.
  Závažnost: Nízká
  Náročnost: Vysoká
  Vyřešeno: Ne
  Návrh řešení: Zobrazit počet relevantních notifikací u odkazu na předmět v postranním panelu by zahrnovalo pouze sestrojení poměrně jednoduchého dotazu do databáze a následné zobrazení získané informace – jeden z atributů notifikace je totiž kód předmětu. Pro ostatní typy objektů jako jsou týmy, odevzdání a podobně by bylo nutné upravit databázové schéma, aby byla informace o relevantním objektu dostupná, momentálně notifikace totiž udržují pouze identifikátor uživatele, kód předmětu, text notifikace a URL k přesměrování uživatele po kliknutí na notifikaci.

- **Popis:** Vyučujícímu nebylo jasné, kam bude přesměrován po kliknutí na tlačítko „Nastavení správce“ na stránce nastavení předmětu na úrovni vyučujícího.
  Závažnost: Nízká
  Náročnost: Nízká
  Vyřešeno: Ne
  Návrh řešení: Stránka správcovského nastavení, na kterou tlačítko vede, umožňuje upravovat výchozí konfiguraci předmětu, správcovská oprávnění jednotlivých vyučující a nastavovat vyučující zodpovědné za jednotlivé paralelky. Autorovi práce se nepodařilo vymyslet výstižnější název než „Nastavení správce“. Název by měl lépe reflektovat i další nastavení kromě samotné výchozí konfigurace předmětu.

- **Popis:** Student nemůže vidět požadavky jednotlivých kontrolních bodů, dokud nemá vytvořený tým.
  Závažnost: Nízká
  Náročnost: Střední
  Vyřešeno: Ne
  Návrh řešení: Pro zobrazení například na stránce předmětu není vždy jasné, které kontrolní body by měly být zobrazeny – budou se totiž řídit nastavením vyučujícího zodpovědného za zvolené zadání. Možná by bylo vhodné tyto kontrolní body zobrazit alespoň na stránce detailu zadání.

- **Popis:** Při vysokém počtu projektů v rámci jednoho předmětu je seznam projektů na stránce předmětů příliš dlouhý. Tento problém nastává především v předmětech, kde studenti vypracovávají projekty samostatně nebo ve velmi malých týmech.
  Závažnost: Nízká
  Náročnost: Střední
  Vyřešeno: Ne
  Návrh řešení: Je možné na tuto stránku přidat stránkování tohoto seznamu a případně i vyhledávání. Také se nabízí v případě vysokého počtu projektů jednotlivé položky zobrazovat kompaktnějším způsobem.

- **Popis:** K rozbalení seznamů pro výběr jazyka, semestru nebo role v navigační liště jsou nutná dvě kliknutí.
  Závažnost: Nízká
  Náročnost: Střední
  Vyřešeno: Ne
  Návrh řešení: Bylo by možné šablonu upravit aby byl widget použitelnější, autor práce to však vzhledem k velmi málo časté potřebě měnit tato nastavení nepovažoval za podstatné.

- **Popis:** Vyučující hodnotil formulář hodnocení odevzdání jako nepřehledný.
  Závažnost: Nízká
  Náročnost: Nízká
  Vyřešeno: Ne
  Návrh řešení: Problém by mohla částečně vyřešit úprava textů potvrzovacích tlačítek, částečně pak úprava uspořádání jednotlivých polí formuláře.

- **Popis:** Vyučující by uvítal možnost vynutit uvedení repozitáře při tvorbě týmu.
  Závažnost: Nízká
  Náročnost: Střední
  Vyřešeno: Ne
  Návrh řešení: Atributy polí formuláře lze dynamicky upravovat v kódu příslušné třídy formuláře. Pro zajištění této funkcionality by pak bylo nutné přidat tuto volbu do konfigurace předmětu.

- **Popis:** Vyučující by uvítal možnost hodnocení i jiných úloh v rámci předmětu než jen semestrální práce.
  Závažnost: Nízká

Náročnost: Vysoká
Vyřešeno: Ne
Návrh řešení: Tato funkcionalita je mimo rámec práce. Nejlepším řešením do budoucna by podle autora bylo obousměrné propojení systému s aplikací FIT Klasifikace a následné zobrazování odtud získaných informací.

- **Popis:** Vyučující vnímá jako zbytečnou globální volbu rolí studenta a vyučujícího pro celý systém, když se role vztahuje vždy ke konkrétnímu předmětu v konkrétním semestru.
  Závažnost: Nízká
  Náročnost: Střední
  Vyřešeno: Ne
  Návrh řešení: Záměřem autora bylo odlišit domovskou stránku a postranní panel vyučujícího a studenta. Pokud by stejný názor jako měl testující subjekt časem projevilo více uživatelů, kteří jsou zároveň studenty i vyučujícími, je možné se bez větších problémů globálních rolí zbavit a domovské stránky studenta a vyučujícího spojit do jedné.

- **Popis:** Vyučující by uvítal možnost přidávat do systému vlastní paralelky v případě jejich špatného importu z KOSu.
  Závažnost: Střední
  Náročnost: Vysoká
  Vyřešeno: Ne
  Návrh řešení: Bylo by potřeba připravit příslušné Views, formuláře a logiku zajišťující konzistenci dat. V současném stavu se systém spoléhá na to, že data z KOSu jsou konzistentní. Potřeba vyučujícího vytvářet vlastní paralelky by mohla být částečně vyřešena tím, že byla přidána možnost výběru typu paralelky při importu předmětu z KOSu, jak bylo zmíněno v jendom z předchozích bodů, což by mělo vyřešit problémy s nevhodným rozdělením studentů do paralelek v systému SOS.

- **Popis:** Vyučující by uvítal možnost omezit nabídku zadání pro konkrétního studenta.
  Závažnost: Nízká
  Náročnost: Vysoká
  Vyřešeno: Ne
  Návrh řešení: Tato funkcionalita by pravděpodobně byla důležitá pro předmět BI-PA2, ostatní předměty by ji nejspíše nevyužily. Bylo by nutné kompletně navrhnout principy fungování této funkcionality a následně je implementovat. Principy fungování by se mohly inspirovat sys-

témem ProgTest, který tuto funkcionalitu dle vyjádření vyučujícího má.

Příloha F

Seznam použitých zkratek

API Application Programming Interface

CI/CD Continuous Integration / Continuous Deployment

CSRF Cross-site Request Forgery

CSS Cascading Style Sheets

CSV Comma-separated Values

CVS Concurrent Version System

ČVUT České vysoké učení technické

DIMM Dual In-line Memory Module

FIT Fakulta informačních technologií

FP Funkční požadavek

FURPS Functionality, Usability, Reliability, Performance, Supportability

HTML Hypertext Markup Language

HTTP Hypertext Transfer Protocol

HTTPS Hypertext Transfer Protocol Secure

IP Internet Protocol

JSON JavaScript Object Notation

MPA Multi-page application

MVC Model-View-Controller

MVT Model-View-Template

ORM Object-relational-mapping

PDF Portable Document Format

PEP Python Enhancement Proposal

REST Representational State Transfer

RFC Request for Comments

SPA Single-page application

SQL Structured Query Language

SSH Secure Shell

SSL Secure Sockets Layer

SSO Single Sign-On

TLS Transport Layer Security

UC Use Case

URI Uniform Resource Identifier

URL Uniform Resource Locator

UUID Universally Unique Identifier

WSGI Web Server Gateway Interface

XML Extensible Markup Language

XSS Cross-site Scripting

Příloha G

Obsah přiloženého paměťového média

Mimo paměťové médium se veškeré zdrojové kódy se také nacházejí v GitLab repozitáři dostupném na adrese `https://gitlab.fit.cvut.cz/pavlutom/sos`. Do repozitáře byl udělen přístup vedoucímu práce Ing. Jiřímu Hunkovi a oponentovi práce Ing. Janu Matouškovi. Na požádání bude přístup udělen i dalším osobám.

Audiovizuální záznamy z uživatelského testování se mimo paměťové médium nacházejí i na fakultním úložišti OneDrive a jsou dostupné na adrese `https://campuscvut-my.sharepoint.com/:f:/g/personal/pavlutom_cvut_cz/Epp-wLlRtqRKvqs7Lx5i_ncBN2_AXlyLpmQRIsZe_bsaBA?e=NCYhXX`. Do sdíleného adresáře obsahujícího záznamy byl taktéž udělen přístup vedoucímu práce a oponentovi práce a na požádání bude udělen i dalším osobám.

- `readme.txt` ..... stručný popis obsahu paměťového média
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF
  - `thesis.ps` ..... text práce ve formátu PS
- `recordings` ..... audiovizuální záznamy z uživatelského testování